SIAM J. MATRIX ANAL. APPL.
Vol. 38, No. 4, pp. 1454–1485

PRACTICAL SKETCHING ALGORITHMS FOR LOW-RANK MATRIX APPROXIMATION*

JOEL A. TROPP†, ALP YURTSEVER‡, MADELEINE UDELL§, AND VOLKAN CEVHER‡

Abstract. This paper describes a suite of algorithms for constructing low-rank approximations of an input matrix from a random linear image, or *sketch*, of the matrix. These methods can preserve structural properties of the input matrix, such as positive-semidefiniteness, and they can produce approximations with a user-specified rank. The algorithms are simple, accurate, numerically stable, and provably correct. Moreover, each method is accompanied by an informative error bound that allows users to select parameters a priori to achieve a given approximation quality. These claims are supported by numerical experiments with real and synthetic data.

Key words. dimension reduction, matrix approximation, numerical linear algebra, randomized algorithm, single-pass algorithm, sketching, streaming algorithm, subspace embedding

AMS subject classifications. Primary, 65F30; Secondary, 68W20

DOI. 10.1137/17M1111590

1. Motivation.

This paper presents a framework for computing structured low-rank approximations of a matrix from a *sketch*, which is a random low-dimensional linear image of the matrix. Our goal is to develop simple, practical algorithms that can serve as reliable modules in other applications. The methods apply for the real field ($\mathbb{F} = \mathbb{R}$) and for the complex field ($\mathbb{F} = \mathbb{C}$).

1.1. Low-rank matrix approximation.

Suppose that $\boldsymbol{A} \in \mathbb{F}^{m\times n}$ is an arbitrary matrix. Let r be a target rank parameter where $r \ll \min\{m, n\}$. The computational problem is to produce a low-rank approximation $\hat{\boldsymbol{A}}$ of $\boldsymbol{A}$ whose error is comparable to a best rank-r approximation:

$$\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}} \approx \min_{\mathrm{rank}(\boldsymbol{B})\le r} \|\boldsymbol{A} - \boldsymbol{B}\|_{\mathrm{F}}. \tag{1.1}$$

The notation $\|\cdot\|_{\mathrm{F}}$ refers to the Frobenius norm. We explicitly allow the rank of $\hat{\boldsymbol{A}}$ to exceed r because we can obtain more accurate approximations of this form, and the precise rank of $\hat{\boldsymbol{A}}$ is unimportant in many applications. There has been extensive research on randomized algorithms for (1.1); see Halko, Martinsson, and Tropp [19].

1.2. Sketching.

Here is the twist. Imagine that our interactions with the matrix $\boldsymbol{A}$ are severely constrained in the following way. We construct a linear map $\mathcal{L} : \mathbb{F}^{m\times n} \to \mathbb{F}^d$ that does not depend on the matrix $\boldsymbol{A}$. Our only mechanism for collecting

*Received by the editors January 17, 2017; accepted for publication (in revised form) by August 18, 2017; published electronically December 6, 2017.
http://www.siam.org/journals/simax/38-4/M111159.html
Funding: The work of the first and third authors was supported in part by ONR award N00014-11-1002 and the Gordon & Betty Moore Foundation. The work of the third author was also supported in part by DARPA award FA8750-17-2-0101. The work of the second and fourth authors was supported in part by the European Commission under the ERC Future Proof grant and grants SNF 200021-146750, and SNF CRSII2-147633.

†Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125-5000 (jtropp@cms.caltech.edu).
‡École Polytechnique Fédéral de Lausanne, Lausanne 1015, Switzerland (alp.yurtsever@epfl.ch, volkan.cevher@epfl.ch).
§Cornell University, Ithaca, NY 14853 (mru8@cornell.edu).

data S about $\boldsymbol{A}$ is to apply the linear map $\mathcal{L}$:

$$\mathsf{S} := \mathcal{L}(\boldsymbol{A}) \in \mathbb{F}^d. \tag{1.2}$$

We refer to S as a *sketch* of the matrix, and $\mathcal{L}$ is called a *sketching map*. The number d is called the *dimension* or *size* of the sketch.

The challenge is to make the sketch as small as possible while collecting enough information to approximate the matrix accurately. In particular, we want the sketch dimension d to be much smaller than the total dimension mn of the matrix $\boldsymbol{A}$. As a consequence, the sketching map $\mathcal{L}$ has a substantial null space. Therefore, it is natural to draw the sketching map *at random* so that we are likely to extract useful information from any fixed input matrix.

1.3. Why sketch? There are a number of situations where the sketching model (1.2) is a natural mechanism for acquiring data about an input matrix.

First, imagine that $\boldsymbol{A}$ is a huge matrix that can only be stored outside of core memory. The cost of data transfer may be substantial enough that we can only afford to read the matrix into core memory once [19, sect. 5.5]. We can build a sketch as we scan through the matrix. Other types of algorithms for this problem appear in [15, 16].

Second, there are applications where the columns of the matrix $\boldsymbol{A}$ are revealed one at a time, and we must be able to compute an approximation at any instant. One approach is to maintain a sketch that is updated when a new column arrives. Other types of algorithms for this problem appear in [4, 21].

Third, we may encounter a setting where the matrix $\boldsymbol{A}$ is presented as a sum of ordered updates:

$$\boldsymbol{A} = \boldsymbol{H}_1 + \boldsymbol{H}_2 + \boldsymbol{H}_3 + \boldsymbol{H}_4 + \cdots. \tag{1.3}$$

We must discard each innovation $\boldsymbol{H}_i$ after it is processed [9, 34]. In this case, the random linear sketch (1.2) is more or less the only way to maintain a representation of $\boldsymbol{A}$ through an arbitrary sequence of updates [23]. Our research was motivated by a variant [36] of the model (1.3); see subsection 3.8.

1.4. Overview of algorithms. Let us summarize our basic approach to sketching and low-rank approximation of a matrix. Fix a target rank r and an input matrix $\boldsymbol{A} \in \mathbb{F}^{m\times n}$. Select sketch size parameters k and ℓ. Draw and fix independent standard normal matrices $\boldsymbol{\Omega} \in \mathbb{F}^{n\times k}$ and $\boldsymbol{\Psi} \in \mathbb{F}^{\ell\times m}$; see Definition 2.1. We realize the randomized linear sketch (1.2) via left and right matrix multiplication:

$$\boldsymbol{Y} := \boldsymbol{A\Omega} \quad \text{and} \quad \boldsymbol{W} := \boldsymbol{\Psi A}. \tag{1.4}$$

We can store the random matrices and the sketch using $(k+\ell)(m+n)$ scalars. The arithmetic cost of forming the sketch is $\Theta((k+\ell)mn)$ floating-point operations (flops) for a general matrix $\boldsymbol{A}$.

Given the random matrices $(\boldsymbol{\Omega}, \boldsymbol{\Psi})$ and the sketch $(\boldsymbol{Y}, \boldsymbol{W})$, we compute an approximation $\hat{\boldsymbol{A}}$ in three steps:

1. Form an orthogonal–triangular factorization $\boldsymbol{Y} =: \boldsymbol{QR}$, where $\boldsymbol{Q} \in \mathbb{F}^{m\times k}$.
2. Solve a least-squares problem to obtain $\boldsymbol{X} := (\boldsymbol{\Psi Q})^\dagger \boldsymbol{W} \in \mathbb{F}^{k\times n}$.
3. Construct the rank-k approximation $\hat{\boldsymbol{A}} := \boldsymbol{QX}$.

The total cost of this computation is $\Theta(kl(m+n))$ flops. See subsection 4.2 for the intuition behind this approach.

Now, suppose that we set the sketch size parameters $k = 2r + 1$ and $\ell = 4r + 2$. For this choice, Theorem 4.3 yields the error bound

$$\mathbb{E}\,\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}} \leq 2 \cdot \min_{\mathrm{rank}(\boldsymbol{B}) \leq r} \|\boldsymbol{A} - \boldsymbol{B}\|_{\mathrm{F}}.$$

In other words, we typically obtain an approximation with rank $\approx 2r$ whose error lies within twice the optimal rank-r error! Moreover, the total storage cost is about $6r(m + n)$, which is comparable to the number of degrees of freedom in an $m \times n$ matrix with rank r, so the sketch size cannot be reduced substantially.

1.5. Our contributions. This paper presents a systematic treatment of sketching algorithms for low-rank approximation of a matrix. All of the methods rely on the simple sketch (1.4) of the input matrix (subsection 3.5). The main algorithm uses this sketch to compute a high-quality low-rank approximation $\hat{\boldsymbol{A}}$ of the input matrix (Algorithm 4). We prove that this method automatically takes advantage of spectral decay in the input matrix (Theorem 4.3); this result is new.

We also explain how to compute approximations with additional structure—such as symmetry, positive semidefiniteness, or fixed rank—by projecting the initial low-rank approximation onto the family of structured matrices (sections 5 and 6). This approach ensures that the structured approximations also exploit spectral decay (Fact 5.1 and Proposition 6.1). In the sketching context, this idea is new.

Each algorithm is accompanied by an informative error bound that provides a good description of its actual behavior. As a consequence, we can offer the first concrete guidance on algorithm parameters for various types of input matrices (subsection 4.5), and we can implement the methods with confidence. We also include pseudocode and an accounting of computational costs.

The paper includes a collection of numerical experiments (section 7). This work demonstrates that the recommended algorithms can significantly outperform alternative methods, especially when the input matrix has spectral decay. The empirical work also confirms our guidance on parameter choices.

Our technical report [32] contains some more error bounds for the reconstruction algorithms. It also documents additional numerical experiments.

1.6. Limitations. The algorithms in this paper are not designed for all low-rank matrix approximation problems. They are specifically intended for environments where we can only make a single pass over the input matrix or where the data matrix is presented as a stream of linear updates. When it is possible to make multiple passes over the input matrix, we recommend the low-rank approximation algorithms documented in [19]. Multipass methods are significantly more accurate because they drive the error of the low-rank approximation down to the optimal low-rank approximation error exponentially fast in the number of passes.

1.7. Overview of related work. Randomized algorithms for matrix approximation date back to research [30, 17] in theoretical computer science (TCS) in the late 1990s. Starting around 2004, this work inspired numerical analysts to develop practical algorithms for matrix approximation and related problems [26]. See the paper [19, sect. 2] for a comprehensive historical discussion. The surveys [25, 34] provide more details about the development of these ideas within the TCS literature.

1.7.1. Sketching algorithms for matrix approximation. To the best of our knowledge, the first sketching algorithm for low-rank matrix approximation appears in Woolfe et al. [35, sect. 5.2]. Their primary motivation was to compute a low-rank

matrix approximation faster than any classical algorithm, rather than to work under the constraints of a sketching model. A variant of their approach is outlined in [19, sect. 5.5].

Clarkson and Woodruff [9] explicitly frame the question of how to perform numerical linear algebra tasks under the sketching model (1.2). Among other things, they develop algorithms and lower bounds for low-rank matrix approximation. Some of the methods that we recommend are algebraically—but not numerically—equivalent to formulas [9, Thms. 4.7 and 4.8] that they propose. Their work focuses on obtaining a priori error bounds. In contrast, we also aim to help users implement the methods, choose parameters, and obtain good empirical performance in practice. Additional details appear throughout our presentation.

There are many subsequent theoretical papers on sketching algorithms for low-rank matrix approximation, including [34, 12, 6]. This line of research exploits a variety of tricks to obtain algorithms that, theoretically, attain better asymptotic upper bounds on computational resource usage. Subsection 7.3 contains a representative selection of these methods and their guarantees.

1.7.2. Note added in press. When we wrote this paper, the literature did not contain sketching methods tailored for symmetric or positive-semidefinite (psd) matrix approximation. A theoretical paper [8] on algorithms for low-rank approximation of a sparse psd matrix was released after our work appeared.

1.7.3. Error bounds. Almost all previous papers in this area have centered on the following problem. Let $\boldsymbol{A} \in \mathbb{F}^{m \times n}$ be an input matrix, let r be a target rank, and let $\varepsilon > 0$ be an error tolerance. Given a randomized linear sketch (1.2) of the input matrix, produce a rank-r approximation $\hat{\boldsymbol{A}}_{\text{eps}}$ that satisfies

$$\|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\text{eps}}\|_{\text{F}}^2 \leq (1+\varepsilon) \cdot \min_{\text{rank}(\boldsymbol{B}) \leq r} \|\boldsymbol{A} - \boldsymbol{B}\|_{\text{F}}^2 \quad \text{with high probability.} \tag{1.5}$$

To achieve (1.5) for a general input, the sketch must have dimension $\Omega(r(m+n)/\varepsilon)$ [9, Thm. 4.10]. Furthermore, the analogous error bound for the spectral norm cannot be achieved for all input matrices under the sketching model [34, Chap. 6.2]. Nevertheless, Gu [18, Thm. 3.4] has observed that (1.5) implies a weak error bound in the spectral norm.

Li et al. [22, App.] caution that the guarantee (1.5) is often vacuous. For example, we frequently encounter matrices for which the Frobenius-norm error of an optimal rank-r approximation is larger than the Frobenius norm of the approximation itself. In other settings, it may be necessary to compute an approximation with very high accuracy. Either way, ε must be tiny before the bound (1.5) sufficiently constrains the approximation error. For a general input matrix, to achieve a small value of ε, the sketch size must be exorbitant. We tackle this issue by providing alternative error estimates (e.g., Theorem 4.3) that yield big improvements for most examples.

1.7.4. Questions.... Our aim is to address questions that arise when one attempts to use sketching algorithms in practice. For instance, how do we implement these methods? Are they numerically stable? How should algorithm parameters depend on the input matrix? What is the right way to preserve structural properties? Which methods produce the best approximations in practice? How small an approximation error can we actually achieve? Does existing theoretical analysis predict performance? Can we obtain error bounds that are more illuminating than (1.5)? These questions have often been neglected in the literature.

Our empirical study (section 7) highlights the importance of this inquiry. Surprisingly, numerical experiments reveal that the pursuit of theoretical metrics has been counterproductive. More recent algorithms often perform worse in practice, even though—in principle—they offer better performance guarantees.

2. Background. In this section, we collect notation and conventions, as well as some background on random matrices.

2.1. Notation and conventions. We write $\mathbb{F}$ for the scalar field, which is either $\mathbb{R}$ or $\mathbb{C}$. The letter $\mathbf{I}$ signifies the identity matrix; its dimensions are determined by context. The asterisk * refers to the (conjugate) transpose operation on vectors and matrices. The dagger † is the Moore–Penrose pseudoinverse. The symbol $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm.

The expression "$\boldsymbol{M}$ has rank r" and its variants mean that the rank of $\boldsymbol{M}$ does not exceed r. The symbol $[\![\boldsymbol{M}]\!]_r$ represents an optimal rank-r approximation of $\boldsymbol{M}$ with respect to the Frobenius norm; this approximation need not be unique [20, sect. 6].

It is valuable to introduce notation for the error incurred by a best rank-r approximation in the Frobenius norm. For each natural number j, we define the *jth tail energy*

$$\tau_j^2(\boldsymbol{A}) := \min_{\mathrm{rank}(\boldsymbol{B})<j} \|\boldsymbol{A}-\boldsymbol{B}\|_{\mathrm{F}}^2 = \sum\nolimits_{i\geq j} \sigma_i^2(\boldsymbol{A}). \tag{2.1}$$

We have written $\sigma_i(\boldsymbol{A})$ for the ith largest singular value of $\boldsymbol{A}$. The equality follows from the Eckart–Young theorem; for example, see [20, sect. 6].

The symbol $\mathbb{E}$ denotes expectation with respect to all random variables. For a given random variable Z, we write $\mathbb{E}_Z$ to denote expectation with respect to the randomness in Z only. Nonlinear functions bind before the expectation.

In the description of algorithms in the text, we primarily use standard mathematical notation. In the pseudocode, we rely on some MATLAB R2017a functions in an effort to make the presentation more concise.

We use the computer science interpretation of $\Theta(\cdot)$ to refer to the class of functions whose growth is bounded above and below up to a constant.

2.2. Standard normal matrices. Let us define an ensemble of random matrices that plays a central role in this work.

DEFINITION 2.1 (standard normal matrix). *A matrix $\boldsymbol{G} \in \mathbb{R}^{m\times n}$ has the real standard normal distribution if the entries form an independent family of standard normal random variables (i.e., Gaussian with mean zero and variance one).*

A matrix $\boldsymbol{G} \in \mathbb{C}^{m\times n}$ has the complex standard normal distribution if it has the form $\boldsymbol{G} = \boldsymbol{G}_1 + \mathrm{i}\boldsymbol{G}_2$, where $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are independent, real standard normal matrices.

Standard normal matrices are also known as Gaussian matrices.

We introduce numbers α and β that reflect the field over which the random matrix is defined:

$$\alpha := \alpha(\mathbb{F}) := \begin{cases} 1, & \mathbb{F} = \mathbb{R}, \\ 0, & \mathbb{F} = \mathbb{C}, \end{cases} \quad \text{and} \quad \beta := \beta(\mathbb{F}) := \begin{cases} 1, & \mathbb{F} = \mathbb{R}, \\ 2, & \mathbb{F} = \mathbb{C}. \end{cases} \tag{2.2}$$

This notation allows us to treat the real and complex cases simultaneously. The number β is a standard parameter in random matrix theory.

Finally, we introduce notation to help make our theorem statements more succinct:

$$f(s,t) := \frac{s}{t-s-\alpha} \quad \text{for integers that satisfy } t > s+\alpha > \alpha. \tag{2.3}$$

Observe that the function $f(s,\cdot)$ is decreasing, with range $(0,s]$.

3. Sketching the input matrix. First, we discuss how to collect enough data about an input matrix to compute a low-rank approximation. We summarize the matrix by multiplying it on the right and the left by random test matrices. The dimension and distribution of these random test matrices together determine the potential accuracy of the approximation.

3.1. The input matrix. Let $\boldsymbol{A} \in \mathbb{F}^{m\times n}$ be a matrix that we wish to approximate. Our algorithms work regardless of the relative dimensions of $\boldsymbol{A}$, but there may sometimes be small benefits if we apply them to $\boldsymbol{A}^*$ instead.

3.2. The target rank. Let r be a target rank parameter with $1 \le r \le \min\{m,n\}$. We aim to construct a low-rank approximation of $\boldsymbol{A}$ whose error is close to the optimal rank-r error. We explicitly allow approximations with rank somewhat larger than r because they may be significantly more accurate.

Under the sketching model (1.2), the practitioner must use prior knowledge about the input matrix $\boldsymbol{A}$ to determine a target rank r that will result in satisfactory error guarantees. This decision is outside the scope of our work.

3.3. Parameters for the sketch. The sketch consists of two parts: a summary of the range of $\boldsymbol{A}$ and a summary of the co-range. The parameter k controls the size of the range sketch, and the parameter ℓ controls the size of the co-range sketch. They should satisfy the conditions

$$r \le k \le \ell \quad \text{and} \quad k \le n \quad \text{and} \quad \ell \le m. \tag{3.1}$$

We often choose $k \approx r$ and $\ell \approx k$. See (4.6) and subsection 4.5 below.

The parameters k and ℓ do not play symmetrical roles. We need $\ell \ge k$ to ensure that a certain $\ell \times k$ matrix has full column rank. Larger values of both k and ℓ result in better approximations at the cost of more storage and arithmetic. These trade-offs are quantified in what follows.

3.4. The test matrices. To form the sketch of the input matrix, we draw and fix two (random) test matrices:

$$\boldsymbol{\Omega} \in \mathbb{F}^{n\times k} \quad \text{and} \quad \boldsymbol{\Psi} \in \mathbb{F}^{\ell\times m}. \tag{3.2}$$

This paper contains a detailed analysis of the case where the test matrices are statistically independent and follow the standard normal distribution. Subsection 3.9 describes other potential distributions for the test matrices. We always state when we are making distributional assumptions on the test matrices.

3.5. The sketch. The sketch of $\boldsymbol{A} \in \mathbb{F}^{m\times n}$ consists of two matrices:

$$\boldsymbol{Y} := \boldsymbol{A}\boldsymbol{\Omega} \in \mathbb{F}^{m\times k} \quad \text{and} \quad \boldsymbol{W} := \boldsymbol{\Psi}\boldsymbol{A} \in \mathbb{F}^{\ell\times n}. \tag{3.3}$$

The matrix $\boldsymbol{Y}$ collects information about the action of $\boldsymbol{A}$, while the matrix $\boldsymbol{W}$ collects information about the action of $\boldsymbol{A}^*$. Both parts are necessary.

Algorithm 1 *Sketch for Low-Rank Approximation.* Implements (3.2) and (3.3).

Require: Input matrix $\boldsymbol{A} \in \mathbb{F}^{m \times n}$; sketch size parameters $k \leq \ell$
Ensure: Constructs test matrices $\boldsymbol{\Omega} \in \mathbb{F}^{n \times k}$ and $\boldsymbol{\Psi} \in \mathbb{F}^{\ell \times m}$, range sketch $\boldsymbol{Y} = \boldsymbol{A\Omega} \in \mathbb{F}^{m \times k}$, and co-range sketch $\boldsymbol{W} = \boldsymbol{\Psi A} \in \mathbb{F}^{\ell \times n}$ as private variables

private: $\boldsymbol{\Omega}, \boldsymbol{\Psi}, \boldsymbol{Y}, \boldsymbol{W}$ ▷ Internal variables for SKETCH object ▷ Accessible to all SKETCH methods

function SKETCH($\boldsymbol{A}; k, \ell$) ▷ Constructor
  if $\mathbb{F} = \mathbb{R}$ **then**
    $\boldsymbol{\Omega} \leftarrow \texttt{randn}(n, k)$
    $\boldsymbol{\Psi} \leftarrow \texttt{randn}(\ell, m)$
  if $\mathbb{F} = \mathbb{C}$ **then**
    $\boldsymbol{\Omega} \leftarrow \texttt{randn}(n, k) + \texttt{i}\,\texttt{randn}(n, k)$
    $\boldsymbol{\Psi} \leftarrow \texttt{randn}(\ell, m) + \texttt{i}\,\texttt{randn}(\ell, m)$
  $\boldsymbol{\Omega} \leftarrow \texttt{orth}(\boldsymbol{\Omega})$ ▷ (optional) Improve numerical stability
  $\boldsymbol{\Psi}^* \leftarrow \texttt{orth}(\boldsymbol{\Psi}^*)$ ▷ (optional) Improve numerical stability
  $\boldsymbol{Y} \leftarrow \boldsymbol{A\Omega}$
  $\boldsymbol{W} \leftarrow \boldsymbol{\Psi A}$

Remark 3.1 (prior work). The matrix sketching algorithms that appear in [35, sect. 5.2], [9, Thm. 4.9], [19, sect. 5.5], and [34, Thm. 4.3] all involve a sketch of the form (3.3). In contrast, the most recent approaches ([6, sect. 6.1.2] and [33, sect. 3]) use more complicated sketches; see subsection 7.3.2.

3.6. The sketch as an abstract data type. We present the sketch as an abstract data type using ideas from object-oriented programming. SKETCH is an object that contains information about a specific matrix $\boldsymbol{A}$. The test matrices $(\boldsymbol{\Omega}, \boldsymbol{\Psi})$ and the sketch matrices $(\boldsymbol{Y}, \boldsymbol{W})$ are private variables that are only accessible to the SKETCH methods. A user interacts with the SKETCH object by initializing it with a specific matrix and by applying linear updates. The user can query the SKETCH object to obtain an approximation of the matrix $\boldsymbol{A}$ with specific properties. The individual algorithms described in this paper are all methods that belong to the SKETCH object.

3.7. Initializing the sketch and its costs. See Algorithm 1 for pseudocode that implements the sketching procedure (3.2) and (3.3) with either standard normal test matrices (default) or random orthonormal test matrices (optional steps). Note that the orthogonalization step requires additional arithmetic and communication.

The storage cost for the sketch $(\boldsymbol{Y}, \boldsymbol{W})$ is $mk + \ell n$ floating-point numbers in the field $\mathbb{F}$. The storage cost for two standard normal test matrices is $nk + \ell m$ floating-point numbers in $\mathbb{F}$. Some other types of test matrices $(\boldsymbol{\Omega}, \boldsymbol{\Psi})$ have lower storage costs, but the sketch $(\boldsymbol{Y}, \boldsymbol{W})$ remains the same size.

For standard normal test matrices, the arithmetic cost of forming the sketch (3.3) is $\Theta((k + \ell)mn)$ flops when $\boldsymbol{A}$ is dense. If $\boldsymbol{A}$ is sparse, the cost is proportional to the number $\texttt{nnz}(\boldsymbol{A})$ of nonzero entries: $\Theta((k + \ell)\,\texttt{nnz}(\boldsymbol{A}))$ flops. Other types of test matrices sometimes yield lower arithmetic costs.

Algorithm 2 *Linear Update to Sketch.* Implements (3.4).

Require: Update matrix $\boldsymbol{H} \in \mathbb{F}^{m \times n}$; scalars $\theta, \eta \in \mathbb{F}$
Ensure: Modifies sketch $(\boldsymbol{Y}, \boldsymbol{W})$ to reflect linear update $\boldsymbol{A} \leftarrow \theta\boldsymbol{A} + \eta\boldsymbol{H}$

```
function SketchLinearUpdate(H; θ, η)
    Y ← θY + ηHΩ          ▷ Linear update to range sketch
    W ← θW + ηΨH          ▷ Linear update to co-range sketch
```

3.8. Processing linear updates. The sketching model (3.3) supports a linear update that is more general than (1.3). Suppose the input matrix $\boldsymbol{A}$ is modified as

$$\boldsymbol{A} \leftarrow \theta\boldsymbol{A} + \eta\boldsymbol{H}, \quad \text{where } \theta, \eta \in \mathbb{F}.$$

Then we update the sketch (3.3) via the rule

$$\boldsymbol{Y} \leftarrow \theta\boldsymbol{Y} + \eta\boldsymbol{H}\boldsymbol{\Omega} \quad \text{and} \quad \boldsymbol{W} \leftarrow \theta\boldsymbol{W} + \eta\boldsymbol{\Psi}\boldsymbol{H}. \tag{3.4}$$

The precise cost of the computation depends on the structure of $\boldsymbol{H}$. See Algorithm 2 for the pseudocode. This type of update is crucial for certain applications [36].

3.9. Choosing the distribution of the test matrices. Our analysis is specialized to the case where the test matrices $\boldsymbol{\Omega}$ and $\boldsymbol{\Psi}$ are standard normal so that we can obtain highly informative error bounds.

But there are potential benefits from implementing the sketch using test matrices drawn from another distribution. The choice of distribution leads to some tradeoffs in the range of permissible parameters; the costs of randomness, arithmetic, and communication to generate the test matrices; the storage costs for the test matrices and the sketch; the arithmetic costs for sketching and updates; the numerical stability of matrix approximation algorithms; and the quality of a priori error bounds.

Let us list some of the contending distributions along with background references. We have ranked these in decreasing order of reliability.

- **Orthonormal.** The optional steps in Algorithm 1 generate matrices $\boldsymbol{\Omega}$ and $\boldsymbol{\Psi}^*$ with orthonormal columns that span uniformly random subspaces of dimension k and ℓ. When k and ℓ are very large, these matrices result in smaller errors and better numerical stability than Gaussians [14, 19].
- **Gaussian.** Following [26, 19], this paper focuses on test matrices with the standard normal distribution. Benefits include excellent practical performance and accurate a priori error bounds.
- **Rademacher.** These test matrices have independent Rademacher[1] entries. Their behavior is similar to Gaussian test matrices, but there are minor improvements in the cost of storage and arithmetic, as well as the amount of randomness required. For example, see [9].
- **Subsampled randomized Fourier transform (SRFT).** These test matrices take the form

$$\boldsymbol{\Omega} = \boldsymbol{D}_1\boldsymbol{F}_1\boldsymbol{P}_1 \quad \text{and} \quad \boldsymbol{\Psi} = \boldsymbol{P}_2\boldsymbol{F}_2^*\boldsymbol{D}_2, \tag{3.5}$$

[1] A Rademacher random variable takes the values ± 1 with equal probability.

where $\boldsymbol{D}_1 \in \mathbb{F}^{n\times n}$ and $\boldsymbol{D}_2 \in \mathbb{F}^{m\times m}$ are diagonal matrices with independent Rademacher entries; $\boldsymbol{F}_1 \in \mathbb{F}^{n\times n}$ and $\boldsymbol{F}_2 \in \mathbb{F}^{m\times m}$ are discrete cosine transform ($\mathbb{F} = \mathbb{R}$) or discrete Fourier transform ($\mathbb{F} = \mathbb{C}$) matrices; and $\boldsymbol{P}_1 \in \mathbb{F}^{n\times k}$ and $\boldsymbol{P}_2 \in \mathbb{F}^{\ell\times m}$ are restrictions onto k and ℓ coordinates, chosen uniformly at random. These matrices work well in practice, they require a modest amount of storage, and they support fast arithmetic. See [1, 35, 2, 19, 31, 5, 13].

- **Ultrasparse Rademacher.** Let s be a sparsity parameter. In each row of $\boldsymbol{\Omega}$ and column of $\boldsymbol{\Psi}$, we place independent Rademacher random variables in s uniformly random locations; the remaining entries of the test matrices are zero. These matrices help control storage, arithmetic, and randomness costs. On the other hand, they are somewhat less reliable. For more details, see [10, 28, 27, 29, 34, 3, 11].

Except for ultrasparse Rademacher matrices, these distributions often behave quite like a Gaussian distribution in practice [19, sect. 7.4]. An exhaustive comparison of distributions for the test matrices is outside the scope of this paper; see [24].

4. Low-rank approximation from the sketch. Suppose that we have acquired a sketch $(\boldsymbol{Y}, \boldsymbol{W})$ of the input matrix $\boldsymbol{A}$, as in (3.2) and (3.3). This section presents the most basic algorithm for computing a low-rank approximation of $\boldsymbol{A}$ from the data in the sketch. This simple approach is similar to earlier proposals; see [35, sect. 5.2], [9, Thm. 4.7], [19, sect. 5.5], and [34, Thm. 4.3, display 1].

We have obtained the first accurate error bound for this method. Our result shows how the spectrum of the input matrix affects the approximation quality. This analysis allows us to make parameter recommendations for specific input matrices.

In section 5, we explain how to refine this algorithm to obtain approximations with additional structure. In section 6, we describe modifications of the procedures that produce approximations with fixed rank and additional structure. Throughout, we maintain the notation of section 3.

4.1. The main algorithm. Our goal is to produce a low-rank approximation of the input matrix $\boldsymbol{A}$ using only the knowledge of the test matrices $(\boldsymbol{\Omega}, \boldsymbol{\Psi})$ and the sketch $(\boldsymbol{Y}, \boldsymbol{W})$. Here is the basic method.

The first step in the procedure is to compute an orthobasis for the range of $\boldsymbol{Y}$ by means of an orthogonal–triangular factorization:

$$\boldsymbol{Y} =: \boldsymbol{Q}\boldsymbol{R}, \quad \text{where} \quad \boldsymbol{Q} \in \mathbb{F}^{m\times k}. \tag{4.1}$$

The matrix $\boldsymbol{Q}$ has orthonormal columns; we discard the triangular matrix $\boldsymbol{R}$. The second step uses the co-range sketch $\boldsymbol{W}$ to form the matrix

$$\boldsymbol{X} := (\boldsymbol{\Psi}\boldsymbol{Q})^{\dagger}\boldsymbol{W} \in \mathbb{F}^{k\times n}. \tag{4.2}$$

The random matrix $\boldsymbol{\Psi}\boldsymbol{Q} \in \mathbb{F}^{\ell\times k}$ is very well-conditioned when $\ell \gg k$, so we can perform this computation accurately by solving a least-squares problem. We report the rank-k approximation

$$\hat{\boldsymbol{A}} := \boldsymbol{Q}\boldsymbol{X} \in \mathbb{F}^{m\times n}, \quad \text{where} \quad \boldsymbol{Q} \in \mathbb{F}^{m\times k} \quad \text{and} \quad \boldsymbol{X} \in \mathbb{F}^{k\times n}. \tag{4.3}$$

The factors $\boldsymbol{Q}$ and $\boldsymbol{X}$ are defined in (4.1) and (4.2).

Remark 4.1 (prior work). The approximation $\hat{\boldsymbol{A}}$ is algebraically, but not numerically, equivalent with the approximation that appears in Clarkson and Woodruff [9, Thm. 4.7]; see also [34, Thm. 4.3, display 1]. Our formulation improves upon theirs by avoiding a badly conditioned least-squares problem.

Algorithm 3 *Simplest Low-Rank Approximation.* Implements (4.3).

Ensure: For some $q \leq k$, returns factors $\boldsymbol{Q} \in \mathbb{F}^{m \times q}$ with orthonormal columns and $\boldsymbol{X} \in \mathbb{F}^{q \times n}$ that form a rank-q approximation $\hat{\boldsymbol{A}}_{\text{out}} = \boldsymbol{Q}\boldsymbol{X}$ of the sketched matrix

```
function SketchSimpleLowRankApprox( )
    Q ← orth(Y)                          ▷ Orthobasis for range of Y
    X ← (ΨQ)\W                           ▷ Multiply (ΨQ)^† on left side of W
    return (Q, X)
```

Algorithm 4 *Low-Rank Approximation.* Implements (4.3).

Ensure: Returns factors $\boldsymbol{Q} \in \mathbb{F}^{m \times k}$ with orthonormal columns and $\boldsymbol{X} \in \mathbb{F}^{k \times n}$ that form a rank-k approximation $\hat{\boldsymbol{A}}_{\text{out}} = \boldsymbol{Q}\boldsymbol{X}$ of the sketched matrix

```
function SketchLowRankApprox( )
    (Q, ∼) ← qr(Y, 0)                    ▷ Orthobasis for range of Y
    (U, T) ← qr(ΨQ, 0)                   ▷ Orthogonal–triangular factorization
    X ← T^†(U^*W)                        ▷ Apply inverse by back-substitution
    return (Q, X)
```

4.2. Intuition. To motivate the algorithm, we recall a familiar heuristic [19, sect. 1] from randomized linear algebra, which states that

$$\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}. \tag{4.4}$$

Although we would like to form the rank-k approximation $\boldsymbol{Q}(\boldsymbol{Q}^*\boldsymbol{A})$, we cannot compute the factor $\boldsymbol{Q}^*\boldsymbol{A}$ without revisiting the input matrix $\boldsymbol{A}$. Instead, we exploit the information in the co-range sketch $\boldsymbol{W} = \boldsymbol{\Psi}\boldsymbol{A}$. Notice that

$$\boldsymbol{W} = \boldsymbol{\Psi}(\boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}) + \boldsymbol{\Psi}(\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}) \approx (\boldsymbol{\Psi}\boldsymbol{Q})(\boldsymbol{Q}^*\boldsymbol{A}).$$

The heuristic (4.4) justifies dropping the second term. Multiplying on the left by the pseudoinverse $(\boldsymbol{\Psi}\boldsymbol{Q})^\dagger$, we arrive at the relation

$$\boldsymbol{X} = (\boldsymbol{\Psi}\boldsymbol{Q})^\dagger\boldsymbol{W} \approx \boldsymbol{Q}^*\boldsymbol{A}.$$

These considerations suggest that

$$\hat{\boldsymbol{A}} = \boldsymbol{Q}\boldsymbol{X} \approx \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A} \approx \boldsymbol{A}.$$

One of our contributions is to give substance to these nebulae.

Remark 4.2 (prior work). This intuition is inspired by the discussion in [19, sect. 5.5], and it allows us to obtain sharp error bounds. Our approach is quite different from that of [9, Thm. 4.7] or [34, Thm. 4.3].

4.3. Algorithm and costs. Algorithms 3 and 4 give pseudocode for computing the approximation (4.3). The first presentation uses MATLAB functions to abbreviate some of the steps, while the second includes more implementation details. Note that the use of the `orth` command may result in an approximation with rank q for some $q \leq k$, but the quality of the approximation does not change.

Let us summarize the costs of the approximation procedure (4.1)–(4.3), as implemented in Algorithm 4. The algorithm has working storage of $O(k(m+n))$ floating-point numbers. The arithmetic cost is $\Theta(k\ell(m+n))$ flops, which is dominated by the matrix–matrix multiplications. The orthogonalization step and the back-substitution require $\Theta(k^2(m+n))$ flops, which is almost as significant.

4.4. A bound for the Frobenius-norm error. We have established a very accurate error bound for the approximation (4.3) that is implemented in Algorithms 3 and 4. This analysis is one of the key contributions of this paper.

THEOREM 4.3 (low-rank approximation: Frobenius error). *Assume that the sketch size parameters satisfy $\ell > k+\alpha$. Draw random test matrices $\boldsymbol{\Omega} \in \mathbb{F}^{n\times k}$ and $\boldsymbol{\Psi} \in \mathbb{F}^{\ell\times m}$ independently from the standard normal distribution. Then the rank-k approximation $\hat{\boldsymbol{A}}$ obtained from formula* (4.3) *satisfies*

$$\begin{aligned}\mathbb{E}\,\|\boldsymbol{A}-\hat{\boldsymbol{A}}\|_{\mathrm{F}}^2 &\le (1+f(k,\ell))\cdot \min_{\varrho<k-\alpha}(1+f(\varrho,k))\cdot \tau_{\varrho+1}^2(\boldsymbol{A})\\ &= \frac{k}{\ell-k-\alpha}\cdot \min_{\varrho<k-\alpha}\frac{k}{k-\varrho-\alpha}\cdot \tau_{\varrho+1}^2(\boldsymbol{A}).\end{aligned} \tag{4.5}$$

The index ϱ ranges over natural numbers. The quantities $\alpha(\mathbb{R}) := 1$ and $\alpha(\mathbb{C}) := 0$; the function $f(s,t) := s/(t-s-\alpha)$; and the tail energy τ_j^2 is defined in (2.1).

The proof of Theorem 4.3 appears below in Appendix A.3.

To begin to understand Theorem 4.3, it is helpful to consider a specific parameter choice. Let r be the target rank of the approximation, and select

$$k = 2r+\alpha \quad \text{and} \quad \ell = 2k+\alpha. \tag{4.6}$$

For these sketch size parameters, with $\varrho = r$, Theorem 4.3 implies that

$$\mathbb{E}\,\|\boldsymbol{A}-\hat{\boldsymbol{A}}\|_{\mathrm{F}}^2 \le 4\cdot\tau_{r+1}^2(\boldsymbol{A}).$$

In other words, for $k \approx 2r$, we can construct a rank-k approximation of $\boldsymbol{A}$ that has almost the same quality as a best rank-r approximation. This parameter choice balances the sketch size against the quality of approximation.

But the true meaning of Theorem 4.3 lies deeper. The minimum in (4.5) reveals that the approximation (4.3) automatically takes advantage of decay in the tail energy. This fundamental fact explains the strong empirical performance of (4.3) and other approximations derived from it. Our analysis is the first to identify this feature.

Remark 4.4 (prior work). The analysis in [9, Thm. 3.7] shows that $\hat{\boldsymbol{A}}$ achieves a bound of the form (1.5) when the sketch size parameters scale as $k = \Theta(r/\varepsilon)$ and $\ell = \Theta(k/\varepsilon)$. A precise variant of the same statement follows from Theorem 4.3.

Remark 4.5 (high-probability error bound). The expectation bound presented in Theorem 4.3 also describes the typical behavior of the approximation (4.3) because of measure concentration effects. It is possible to develop a high-probability bound using the methods from [19, sect. 10.3].

Remark 4.6 (spectral-norm error bound). It is also possible to develop bounds for the spectral-norm error incurred by the approximation (4.3). These results depend on the decay of both the singular values and the tail energies. See [32, Thm. 4.2].

TABLE 1
Theoretical sketch size parameters. *This table summarizes how to choose the sketch size parameters (k, ℓ) to exploit prior information about the spectrum of the input matrix $\boldsymbol{A}$.*

Problem regime	Notation	Equation
General purpose	$(k_\natural, \ell_\natural)$	(4.9)
Flat spectrum	$(k_\flat, \ell_\flat)$	(4.7) and (4.8)
Decaying spectrum	$(k_\natural, \ell_\natural)$	(4.9)
Rapidly decaying spectrum	$(k_\sharp, \ell_\sharp)$	(4.10)

4.5. Theoretical guidance on the sketch size. Theorem 4.3 is precise enough to predict the performance of the approximation (4.3) for many types of input matrices. As a consequence, we can offer concrete guidance on the best sketch size parameters (k, ℓ) for various applications.

Observe that the storage cost of the sketch (3.3) is directly proportional to the sum $T := k + \ell$ of the sketch size parameters k and ℓ. In this section, we investigate the best way to apportion k and ℓ when we fix the target rank r and the total sketch size T. Throughout this discussion, we assume that $T \geq 2r + 3\alpha + 3$. See Table 1 for a summary of these rules; see subsection 7.5 for an empirical evaluation.

4.5.1. Flat spectrum. First, suppose that the singular values $\sigma_j(\boldsymbol{A})$ of the input matrix $\boldsymbol{A}$ do not decay significantly for $j > r$. This situation occurs, for example, when the input is a rank-r matrix plus white noise.

In this setting, the minimum in (4.5) is likely to occur when $\varrho \approx r$. It is natural to set $\varrho = r$ and to minimize the resulting bound subject to the constraints $k + \ell = T$ and $k > r + \alpha$ and $\ell > k + \alpha$. For $\mathbb{F} = \mathbb{C}$, we obtain the parameter recommendations

$$k_\flat := \max\left\{ r + 1, \ \left\lfloor T \cdot \frac{\sqrt{r(T-r)} - r}{T - 2r} \right\rfloor \right\} \quad \text{and} \quad \ell_\flat := T - k_\flat. \tag{4.7}$$

In the case $\mathbb{F} = \mathbb{R}$, we modify the formula (4.7) so that

$$k_\flat := \max\left\{ r + 2, \left\lfloor (T-1) \cdot \frac{\sqrt{r(T-r-2)(1 - 2/(T-1))} - (r-1)}{T - 2r - 1} \right\rfloor \right\}. \tag{4.8}$$

We omit the routine details behind these calculations.

4.5.2. Decaying spectrum or spectral gap. Suppose that the singular values $\sigma_j(\boldsymbol{A})$ decay at a slow to moderate rate for $j > r$. Alternatively, we may suppose that there is a gap in the singular value spectrum at an index $j > r$.

In this setting, we want to exploit decay in the tail energy by setting $k \gg r$, but we need to ensure that the term $f(k, \ell)$ in (4.5) remains small by setting $\ell \approx 2k + \alpha$. This intuition leads to the parameter recommendations

$$k_\natural := \max\{ r + \alpha + 1, \ \lfloor (T - \alpha)/3 \rfloor \} \quad \text{and} \quad \ell_\natural := T - k_\natural. \tag{4.9}$$

This is the best single choice for handling a range of examples. The parameter recommendation (4.6) is an instance of (4.9) with a minimal value of T.

4.5.3. Rapidly decaying spectrum. Last, assume that the singular values $\sigma_j(\boldsymbol{A})$ decay very quickly for $j > r$. This situation occurs in the application [36] that motivated us to write this paper.

In this setting, we want to exploit decay in the tail energy fully by setting k as large as possible; the benefit outweighs the increase in $f(k, \ell)$ from choosing $\ell = k + \alpha + 1$, the minimum possible value. This intuition leads to the parameter recommendations

$$k_\sharp := \lfloor (T - \alpha - 1)/2 \rfloor \quad \text{and} \quad \ell_\sharp := T - k_\sharp. \tag{4.10}$$

Note that the choice (4.10) is unwise unless the input matrix has sharp spectral decay.

5. Low-rank approximations with convex structure. In many instances, we need to reconstruct an input matrix that has additional structure, such as symmetry or positive-semidefiniteness. The approximation formula (4.3) from section 4 produces an approximation with no special properties aside from a bound on its rank. Therefore, we may have to reform our approximation to instill additional virtues.

In this section, we consider a class of problems where the input matrix belongs to a convex set, and we seek an approximation that belongs to the same set. To accomplish this goal, we replace our initial approximation with the closest point in the convex set. This procedure always improves the Frobenius-norm error.

We address two specific examples: (i) the case where the input matrix is conjugate symmetric, and (ii) the case where the input matrix is positive-semidefinite (psd). In both situations, we must design the algorithm carefully to avoid forming large matrices.

5.1. Projection onto a convex set. Let C be a closed and convex set of matrices in $\mathbb{F}^{m\times n}$. Define the projector $\boldsymbol{\Pi}_C$ onto the set C to be the map

$$\boldsymbol{\Pi}_C : \mathbb{F}^{m\times n} \to C \quad \text{where} \quad \boldsymbol{\Pi}_C(\boldsymbol{M}) := \arg\min \big\{ \|\boldsymbol{C} - \boldsymbol{M}\|_{\mathrm{F}}^2 : \boldsymbol{C} \in C \big\}.$$

The arg min operator returns the matrix $\boldsymbol{C}_\star \in C$ that solves the optimization problem. The solution $\boldsymbol{C}_\star$ is uniquely determined because the squared Frobenius norm is strictly convex and the constraint set C is closed and convex.

5.2. Structure via convex projection. Suppose that the input matrix $\boldsymbol{A}$ belongs to the closed convex set $C \subset \mathbb{F}^{m\times n}$. Let $\hat{\boldsymbol{A}}_{\mathrm{in}} \in \mathbb{F}^{m\times n}$ be an initial approximation of $\boldsymbol{A}$. We can produce a new approximation $\boldsymbol{\Pi}_C(\hat{\boldsymbol{A}}_{\mathrm{in}})$ by projecting the initial approximation onto the constraint set. This procedure always improves the approximation quality in Frobenius norm.

FACT 5.1 (convex structure reduces error). *Let $C \in \mathbb{F}^{m\times n}$ be a closed convex set, and suppose that $\boldsymbol{A} \in C$. For any initial approximation $\hat{\boldsymbol{A}}_{\mathrm{in}} \in \mathbb{F}^{m\times n}$,*

$$\|\boldsymbol{A} - \boldsymbol{\Pi}_C(\hat{\boldsymbol{A}}_{\mathrm{in}})\|_{\mathrm{F}} \le \|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\mathrm{in}}\|_{\mathrm{F}}. \tag{5.1}$$

This result is well known in convex analysis. It follows directly from the first-order optimality conditions [7, sect. 4.2.3] for the Frobenius-norm projection of a matrix onto the set C. We omit the details.

Warning 5.2 (spectral norm). Fact 5.1 does not hold if we replace the Frobenius norm by the spectral norm.

5.3. Low-rank approximation with conjugate symmetry. When the input matrix is conjugate symmetric, it is often critical to produce a conjugate symmetric approximation. We can do so by combining the simple approximation from section 4 with the projection step outlined in subsection 5.1.

5.3.1. Conjugate symmetric projection. Define the set $\mathbb{H}^n(\mathbb{F})$ of conjugate symmetric matrices with dimension n over the field $\mathbb{F}$:

$$\mathbb{H}^n := \mathbb{H}^n(\mathbb{F}) := \{\boldsymbol{C} \in \mathbb{F}^{n\times n} : \boldsymbol{C} = \boldsymbol{C}^*\}.$$

The set $\mathbb{H}^n(\mathbb{F})$ is convex because it forms a real-linear subspace in $\mathbb{F}^{n\times n}$. In what follows, we omit the field $\mathbb{F}$ from the notation unless there is a possibility of confusion.

The projection $\boldsymbol{M}_{\mathrm{sym}}$ of a matrix $\boldsymbol{M} \in \mathbb{F}^{n\times n}$ onto the set $\mathbb{H}^n$ takes the form

$$\boldsymbol{M}_{\mathrm{sym}} := \boldsymbol{\Pi}_{\mathbb{H}^n}(\boldsymbol{M}) = \frac{1}{2}(\boldsymbol{M} + \boldsymbol{M}^*). \tag{5.2}$$

For example, see [20, sect. 2].

5.3.2. Computing a conjugate symmetric approximation. Assume that the input matrix $\boldsymbol{A} \in \mathbb{H}^n$ is conjugate symmetric. Let $\hat{\boldsymbol{A}} := \boldsymbol{QX}$ be an initial rank-k approximation of $\boldsymbol{A}$ obtained from the approximation procedure (4.3). We can form a better Frobenius-norm approximation $\hat{\boldsymbol{A}}_{\mathrm{sym}}$ by projecting $\hat{\boldsymbol{A}}$ onto $\mathbb{H}^n$:

$$\hat{\boldsymbol{A}}_{\mathrm{sym}} := \boldsymbol{\Pi}_{\mathbb{H}^n}(\hat{\boldsymbol{A}}) = \frac{1}{2}(\hat{\boldsymbol{A}} + \hat{\boldsymbol{A}}^*) = \frac{1}{2}(\boldsymbol{QX} + \boldsymbol{X}^*\boldsymbol{Q}^*). \tag{5.3}$$

The second relation follows from (5.2).

In most cases, it is preferable to present the approximation (5.3) in factored form. To do so, we observe that

$$\frac{1}{2}(\boldsymbol{QX} + \boldsymbol{X}^*\boldsymbol{Q}^*) = \frac{1}{2}\begin{bmatrix}\boldsymbol{Q} & \boldsymbol{X}^*\end{bmatrix}\begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \mathbf{I} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{Q} & \boldsymbol{X}^*\end{bmatrix}^*.$$

Concatenate $\boldsymbol{Q}$ and $\boldsymbol{X}^*$, and compute the orthogonal–triangular factorization

$$\begin{bmatrix}\boldsymbol{Q} & \boldsymbol{X}^*\end{bmatrix} =: \boldsymbol{U}\begin{bmatrix}\boldsymbol{T}_1 & \boldsymbol{T}_2\end{bmatrix}, \quad \text{where} \quad \boldsymbol{U} \in \mathbb{F}^{n\times 2k} \text{ and } \boldsymbol{T}_1 \in \mathbb{F}^{2k\times k}. \tag{5.4}$$

Of course, we only need to orthogonalize the k columns of $\boldsymbol{X}^*$, which permits some computational efficiencies. Next, introduce the matrix

$$\boldsymbol{S} := \frac{1}{2}\begin{bmatrix}\boldsymbol{T}_1 & \boldsymbol{T}_2\end{bmatrix}\begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \mathbf{I} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{T}_1 & \boldsymbol{T}_2\end{bmatrix}^* = \frac{1}{2}(\boldsymbol{T}_1\boldsymbol{T}_2^* + \boldsymbol{T}_2\boldsymbol{T}_1^*) \in \mathbb{F}^{2k\times 2k}. \tag{5.5}$$

Combine the last four displays to obtain the rank-$(2k)$ conjugate symmetric approximation

$$\hat{\boldsymbol{A}}_{\mathrm{sym}} = \boldsymbol{USU}^*. \tag{5.6}$$

From this expression, it is easy to obtain other types of factorizations, such as an eigenvalue decomposition, by further processing.

5.3.3. Algorithm, costs, and error. Algorithm 5 contains pseudocode for producing a conjugate symmetric approximation of the form (5.6) from a sketch of the input matrix. One can make this algorithm slightly more efficient by taking advantage of the fact that $\boldsymbol{Q}$ already has orthogonal columns; we omit the details.

For Algorithm 5, the total working storage is $\Theta(kn)$ and the arithmetic cost is $\Theta(k\ell n)$. These costs are dominated by the call to SKETCH.LOWRANKAPPROX.

Combining Theorem 4.3 with Fact 5.1, we have the following bound on the error of the symmetric approximation (5.6), implemented in Algorithm 5. As a consequence, the parameter recommendations from subsection 4.5 are also valid here.

Algorithm 5 *Low-Rank Symmetric Approximation.* Implements (5.6).

Require: Matrix dimensions $m = n$

Ensure: For $q = 2k$, returns factors $\boldsymbol{U} \in \mathbb{F}^{n\times q}$ with orthonormal columns and $\boldsymbol{S} \in \mathbb{H}^q$ that form a rank-q conjugate symmetric approximation $\hat{\boldsymbol{A}}_{\mathrm{out}} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{U}^*$ of the sketched matrix

function SKETCH.LOWRANKSYMAPPROX( )
  $(\boldsymbol{Q}, \boldsymbol{X}) \leftarrow$ LOWRANKAPPROX( ) ▷ Get $\hat{\boldsymbol{A}}_{\mathrm{in}} = \boldsymbol{Q}\boldsymbol{X}$
  $(\boldsymbol{U}, \boldsymbol{T}) \leftarrow \mathtt{qr}([\boldsymbol{Q}, \boldsymbol{X}^*], \mathtt{0})$ ▷ Orthogonal factorization of concatenation
  $\boldsymbol{T}_1 \leftarrow \boldsymbol{T}(:, 1{:}k)$ and $\boldsymbol{T}_2 \leftarrow \boldsymbol{T}(:, (k+1){:}(2k))$ ▷ Extract submatrices
  $\boldsymbol{S} \leftarrow (\boldsymbol{T}_1\boldsymbol{T}_2^* + \boldsymbol{T}_2\boldsymbol{T}_1^*)/2$ ▷ Symmetrize
  return $(\boldsymbol{U}, \boldsymbol{S})$ ▷ Return factors

COROLLARY 5.3 (low-rank symmetric approximation). *Assume that the input matrix $\boldsymbol{A} \in \mathbb{H}^n(\mathbb{F})$ is conjugate symmetric, and assume that the sketch size parameters satisfy $\ell > k + \alpha$. Draw random test matrices $\boldsymbol{\Omega} \in \mathbb{F}^{n\times k}$ and $\boldsymbol{\Psi} \in \mathbb{F}^{\ell\times n}$ independently from the standard normal distribution. Then the rank-$(2k)$ conjugate symmetric approximation $\hat{\boldsymbol{A}}_{\mathrm{sym}}$ produced by* (5.3) *or* (5.6) *satisfies*

$$\mathbb{E}\,\|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\mathrm{sym}}\|_{\mathrm{F}}^2 \le (1 + f(k,\ell)) \cdot \min_{\varrho < k-\alpha} (1 + f(\varrho, k)) \cdot \tau_{\varrho+1}^2(\boldsymbol{A}).$$

The index ϱ ranges over natural numbers. The quantities $\alpha(\mathbb{R}) := 1$ and $\alpha(\mathbb{C}) := 0$; the function $f(s,t) := s/(t - s - \alpha)$; and the tail energy τ_j^2 is defined in (2.1).

5.4. Low-rank positive-semidefinite approximation. We often encounter the problem of approximating a psd matrix. In many situations, it is important to produce an approximation that maintains positivity. Our approach combines the approximation (4.3) from section 4 with the projection step from subsection 5.1.

5.4.1. Positive-semidefinite projection. We introduce the set $\mathbb{H}_+^n(\mathbb{F})$ of psd matrices with dimension n over the field $\mathbb{F}$:

$$\mathbb{H}_+^n := \mathbb{H}_+^n(\mathbb{F}) := \big\{\boldsymbol{C} \in \mathbb{H}^n : \boldsymbol{z}^*\boldsymbol{C}\boldsymbol{z} \ge 0 \text{ for each } \boldsymbol{z} \in \mathbb{F}^n\big\}.$$

The set $\mathbb{H}_+^n(\mathbb{F})$ is convex because it is an intersection of half-spaces. In what follows, we omit the field $\mathbb{F}$ from the notation unless there is a possibility for confusion.

Given a matrix $\boldsymbol{M} \in \mathbb{F}^{n\times n}$, we construct its projection onto the set $\mathbb{H}_+^n$ in three steps. First, form the projection $\boldsymbol{M}_{\mathrm{sym}} := \boldsymbol{\Pi}_{\mathbb{H}^n}(\boldsymbol{M})$ onto the conjugate symmetric matrices, as in (5.2). Second, compute an eigenvalue decomposition $\boldsymbol{M}_{\mathrm{sym}} =: \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^*$. Third, form $\boldsymbol{D}_+$ by zeroing out the negative entries of $\boldsymbol{D}$. Then the projection $\boldsymbol{M}_+$ of the matrix $\boldsymbol{M}$ onto $\mathbb{H}_+^n$ takes the form

$$\boldsymbol{M}_+ := \boldsymbol{\Pi}_{\mathbb{H}_+^n}(\boldsymbol{M}) = \boldsymbol{V}\boldsymbol{D}_+\boldsymbol{V}^*.$$

For example, see [20, sect. 3].

5.4.2. Computing a positive-semidefinite approximation. Assume that the input matrix $\boldsymbol{A} \in \mathbb{H}_+^n$ is psd. Let $\hat{\boldsymbol{A}} := \boldsymbol{Q}\boldsymbol{X}$ be an initial approximation of $\boldsymbol{A}$ obtained from the approximation procedure (4.3). We can form a psd approximation $\hat{\boldsymbol{A}}_+$ by projecting $\hat{\boldsymbol{A}}$ onto the set $\mathbb{H}_+^n$.

Algorithm 6 *Low-Rank Positive-Semidefinite Approximation.* Implements (5.7).

Require: Matrix dimensions $m = n$

Ensure: For $q = 2k$, returns factors $\boldsymbol{U} \in \mathbb{F}^{n \times q}$ with orthonormal columns and nonnegative, diagonal $\boldsymbol{D} \in \mathbb{H}_+^q$ that form a rank-q psd approximation $\hat{\boldsymbol{A}}_{\text{out}} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^*$ of the sketched matrix

```
function Sketch.LowRankPSDApprox( )
    (U, S) ← LowRankSymApprox( )          ▷ Get Â_in = USU*
    (V, D) ← eig(S)                       ▷ Form eigendecomposition
    U ← UV                                ▷ Consolidate orthonormal factors
    D ← max(D, 0)                         ▷ Zero out negative eigenvalues
    return (U, D)
```

To do so, we repeat the computations (5.4) and (5.5) to obtain the symmetric approximation $\hat{\boldsymbol{A}}_{\text{sym}}$ presented in (5.6). Next, form an eigenvalue decomposition of the matrix $\boldsymbol{S}$ given by (5.5):

$$\boldsymbol{S} =: \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^*.$$

In view of (5.6), we obtain an eigenvalue decomposition of $\hat{\boldsymbol{A}}_{\text{sym}}$:

$$\hat{\boldsymbol{A}}_{\text{sym}} = (\boldsymbol{U}\boldsymbol{V})\boldsymbol{D}(\boldsymbol{U}\boldsymbol{V})^*.$$

To obtain the psd approximation $\hat{\boldsymbol{A}}_+$, we simply replace $\boldsymbol{D}$ by its nonnegative part $\boldsymbol{D}_+$ to arrive at the rank-$(2k)$ psd approximation

$$\hat{\boldsymbol{A}}_+ := \boldsymbol{\Pi}_{\mathbb{H}_+^n}(\hat{\boldsymbol{A}}) = (\boldsymbol{U}\boldsymbol{V})\boldsymbol{D}_+(\boldsymbol{U}\boldsymbol{V})^*. \tag{5.7}$$

This formula delivers an approximate eigenvalue decomposition of the input matrix.

5.4.3. Algorithm, costs, and error. Algorithm 6 contains pseudocode for producing a psd approximation of the form (5.7) from a sketch of the input matrix. As in Algorithm 5, some additional efficiencies are possible

The costs of Algorithm 6 are similar with the symmetric approximation method, Algorithm 5. The working storage cost is $\Theta(kn)$, and the arithmetic cost is $\Theta(k\ell n)$.

Combining Theorem 4.3 and Fact 5.1, we obtain a bound on the approximation error identical to Corollary 5.3. We omit the details.

6. Fixed-rank approximations from the sketch. The algorithms in sections 4 and 5 produce high-quality approximations with rank k, but we sometimes need to reduce the rank to match the target rank r. At the same time, we may have to impose additional structure. This section explains how to develop algorithms that produce a rank-r structured approximation.

The technique is conceptually similar to the approach in section 5. We project an initial high-quality approximation onto the set of rank-r matrices. This procedure preserves both conjugate symmetry and the psd property. The analysis in subsection 5.1 does not apply because the set of matrices with fixed rank is not convex. We present a general argument to show that the cost is negligible.

6.1. A general error bound for fixed-rank approximation. If we have a good initial approximation of the input matrix, we can replace this initial approximation by a fixed-rank matrix without increasing the error significantly.

PROPOSITION 6.1 (error for fixed-rank approximation). *Let $\boldsymbol{A} \in \mathbb{F}^{m\times n}$ be a input matrix, and let $\hat{\boldsymbol{A}}_{\mathrm{in}} \in \mathbb{F}^{m\times n}$ be an approximation. For any rank parameter r,*

$$\|\boldsymbol{A} - [\![\hat{\boldsymbol{A}}_{\mathrm{in}}]\!]_r\|_{\mathrm{F}} \le \tau_{r+1}(\boldsymbol{A}) + 2\|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\mathrm{in}}\|_{\mathrm{F}}. \tag{6.1}$$

Recall that $[\![\cdot]\!]_r$ returns a best rank-r approximation with respect to the Frobenius norm.

Proof. Calculate that

$$\begin{aligned}\|\boldsymbol{A} - [\![\hat{\boldsymbol{A}}_{\mathrm{in}}]\!]_r\|_{\mathrm{F}} &\le \|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\mathrm{in}}\|_{\mathrm{F}} + \|\hat{\boldsymbol{A}}_{\mathrm{in}} - [\![\hat{\boldsymbol{A}}_{\mathrm{in}}]\!]_r\|_{\mathrm{F}} \\ &\le \|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\mathrm{in}}\|_{\mathrm{F}} + \|\hat{\boldsymbol{A}}_{\mathrm{in}} - [\![\boldsymbol{A}]\!]_r\|_{\mathrm{F}} \\ &\le 2\|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\mathrm{in}}\|_{\mathrm{F}} + \|\boldsymbol{A} - [\![\boldsymbol{A}]\!]_r\|_{\mathrm{F}}.\end{aligned}$$

The first and last relations are triangle inequalities. To reach the second line, note that $[\![\hat{\boldsymbol{A}}_{\mathrm{in}}]\!]_r$ is a best rank-r approximation of $\hat{\boldsymbol{A}}_{\mathrm{in}}$, while $[\![\boldsymbol{A}]\!]_r$ is an undistinguished rank-r matrix. Finally, identify the tail energy (2.1). ∎

Remark 6.2 (spectral norm). A result analogous to Proposition 6.1 also holds with respect to the spectral norm. The proof is the same.

6.2. Fixed-rank approximation from the sketch. Suppose that we wish to compute a rank-r approximation of the input matrix $\boldsymbol{A} \in \mathbb{F}^{m\times n}$. First, we form an initial approximation $\hat{\boldsymbol{A}} := \boldsymbol{QX}$ using the procedure (4.3). Then we obtain a rank-r approximation $[\![\hat{\boldsymbol{A}}]\!]_r$ of the input matrix by replacing $\hat{\boldsymbol{A}}$ with its best rank-r approximation in the Frobenius norm:

$$[\![\hat{\boldsymbol{A}}]\!]_r = [\![\boldsymbol{QX}]\!]_r. \tag{6.2}$$

We can complete this operation by working directly with the factors. Indeed, suppose that $\boldsymbol{X} = \boldsymbol{U\Sigma V}^*$ is an SVD of $\boldsymbol{X}$. Then $\boldsymbol{QX}$ has an SVD of the form

$$\boldsymbol{QX} = (\boldsymbol{QU})\boldsymbol{\Sigma V}^*.$$

As such, there is also a best rank-r approximation of $\boldsymbol{QX}$ that satisfies

$$[\![\boldsymbol{QX}]\!]_r = (\boldsymbol{QU})[\![\boldsymbol{\Sigma}]\!]_r\boldsymbol{V}^* = \boldsymbol{Q}[\![\boldsymbol{X}]\!]_r.$$

Therefore, the desired rank-r approximation (6.2) can also be expressed as

$$[\![\hat{\boldsymbol{A}}]\!]_r = \boldsymbol{Q}[\![\boldsymbol{X}]\!]_r. \tag{6.3}$$

The formula (6.3) is more computationally efficient than (6.2) because the factor $\boldsymbol{X} \in \mathbb{F}^{k\times n}$ is much smaller than the approximation $\hat{\boldsymbol{A}} \in \mathbb{F}^{m\times n}$.

Remark 6.3 (prior work). The approximation $[\![\hat{\boldsymbol{A}}]\!]_r$ is algebraically, but not numerically, equivalent to a formula proposed by Clarkson and Woodruff [9, Thm. 4.8]. As above, our formulation improves upon theirs by avoiding a badly conditioned least-squares problem.

6.2.1. Algorithm and costs. Algorithm 7 contains pseudocode for computing the fixed-rank approximation (6.3).

The fixed-rank approximation in Algorithm 7 has storage and arithmetic costs on the same order as the simple low-rank approximation (Algorithm 3). Indeed, to compute the truncated SVD and perform the matrix–matrix multiplication, we expend only $\Theta(k^2 n)$ additional flops. Thus, the total working storage is $\Theta(k(m+n))$ numbers and the arithmetic cost is $\Theta(k\ell(m+n))$ flops.

Algorithm 7 *Fixed-Rank Approximation.* Implements (6.3).

Require: Target rank $r \leq k$

Ensure: Returns factors $\boldsymbol{Q} \in \mathbb{F}^{m \times r}$ and $\boldsymbol{V} \in \mathbb{F}^{n \times r}$ with orthonormal columns and nonnegative diagonal $\boldsymbol{\Sigma} \in \mathbb{F}^{r \times r}$ that form a rank-r approximation $\hat{\boldsymbol{A}}_{\mathrm{out}} = \boldsymbol{Q}\boldsymbol{\Sigma}\boldsymbol{V}^*$ of the sketched matrix

```
function Sketch.FixedRankApprox(r)
    (Q, X) ← LowRankApprox( )                ▷ Get Â_in = QX
    (U, Σ, V) ← svds(X, r)                   ▷ Form full SVD and truncate
    Q ← QU                                   ▷ Consolidate orthonormal factors
    return (Q, Σ, V)
```

6.2.2. A bound for the error. We can obtain an error bound for the rank-r approximation (6.3) by combining Theorem 4.3 and Proposition 6.1.

Corollary 6.4 (fixed-rank approximation: Frobenius-norm error). *Assume the sketch size parameters satisfy $k > r + \alpha$ and $\ell > k + \alpha$. Draw random test matrices $\boldsymbol{\Omega} \in \mathbb{F}^{n \times k}$ and $\boldsymbol{\Psi} \in \mathbb{F}^{\ell \times m}$ independently from the standard normal distribution. Then the rank-r approximation $[\![\hat{\boldsymbol{A}}]\!]_r$ obtained from the formula* (6.3) *satisfies*

$$\mathbb{E}\,\|\boldsymbol{A} - [\![\hat{\boldsymbol{A}}]\!]_r\|_{\mathrm{F}} \leq \tau_{r+1}(\boldsymbol{A}) + 2\sqrt{1 + f(k,\ell)} \cdot \min_{\varrho < k - \alpha} \sqrt{1 + f(\varrho, k)} \cdot \tau_{\varrho+1}(\boldsymbol{A}). \tag{6.4}$$

The index ϱ ranges over natural numbers. The quantities $\alpha(\mathbb{R}) := 1$ and $\alpha(\mathbb{C}) := 0$; the function $f(s,t) := s/(t - s - \alpha)$; and the tail energy τ_j^2 is defined in (2.1).

This result indicates that the fixed-rank approximation $[\![\hat{\boldsymbol{A}}]\!]_r$ automatically exploits spectral decay in the input matrix $\boldsymbol{A}$. Moreover, we can still rely on the parameter recommendations from subsection 4.5. Ours is the first theory to provide these benefits.

Remark 6.5 (prior work). The analysis [9, Thm. 4.8] of Clarkson and Woodruff implies that the approximation (6.3) can achieve the bound (1.5) for any $\varepsilon > 0$, provided that $k = \Theta(r/\varepsilon^2)$ and $\ell = \Theta(k/\varepsilon^2)$. It is possible to improve this scaling; see [32, Thm. 5.1].

Remark 6.6 (spectral-norm error bound). It is possible to obtain an error bound for the rank-r approximation (6.3) with respect to the spectral norm by combining [32, Thm. 4.2] and Remark 6.2.

6.3. Fixed-rank conjugate symmetric approximation. Assume that the input matrix $\boldsymbol{A} \in \mathbb{H}^n$ is conjugate symmetric, and we wish to compute a rank-r conjugate symmetric approximation. First, form an initial approximation $\hat{\boldsymbol{A}}_{\mathrm{sym}}$ using the procedure (5.6) in subsection 5.3.2. Then compute an r-truncated eigenvalue decomposition of the matrix $\boldsymbol{S}$ defined in (5.5):

$$\boldsymbol{S} =: \boldsymbol{V}[\![\boldsymbol{D}]\!]_r\boldsymbol{V}^* \;+\; \text{approximation error}.$$

In view of the representation (5.6),

$$[\![\hat{\boldsymbol{A}}_{\mathrm{sym}}]\!]_r = (\boldsymbol{U}\boldsymbol{V})[\![\boldsymbol{D}]\!]_r(\boldsymbol{U}\boldsymbol{V})^*. \tag{6.5}$$

Algorithm 8 contains pseudocode for the fixed-rank approximation (6.5). The total working storage is $\Theta(kn)$, and the arithmetic cost is $\Theta(k\ell n)$.

Algorithm 8 *Fixed-Rank Symmetric Approximation.* Implements (6.5).

Require: Matrix dimensions $m = n$; target rank $r \leq k$

Ensure: Returns factors $\boldsymbol{U} \in \mathbb{F}^{n\times r}$ with orthonormal columns and diagonal $\boldsymbol{D} \in \mathbb{H}^r$ that form a rank-r conjugate symmetric approximation $\hat{\boldsymbol{A}}_{\text{out}} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^*$ of the sketched matrix

function SKETCH.FIXEDRANKSYMAPPROX(r)
  $(\boldsymbol{U}, \boldsymbol{S}) \leftarrow$ LOWRANKSYMAPPROX( ) ▷ Get $\hat{\boldsymbol{A}}_{\text{in}} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{U}^*$
  $(\boldsymbol{V}, \boldsymbol{D}) \leftarrow \texttt{eigs}(\boldsymbol{S}, r, \texttt{'lm'})$ ▷ Truncate full eigendecomposition
  $\boldsymbol{U} \leftarrow \boldsymbol{U}\boldsymbol{V}$ ▷ Consolidate orthonormal factors
  return $(\boldsymbol{U}, \boldsymbol{D})$

Algorithm 9 *Fixed-Rank PSD Approximation.* Implements (6.6).

Require: Matrix dimensions $m = n$; target rank $r \leq k$

Ensure: Returns factors $\boldsymbol{U} \in \mathbb{F}^{n\times r}$ with orthonormal columns and nonnegative, diagonal $\boldsymbol{D} \in \mathbb{H}^r_+$ that form a rank-r psd approximation $\hat{\boldsymbol{A}}_{\text{out}} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^*$ of the sketched matrix

function SKETCH.FIXEDRANKPSDAPPROX(r)
  $(\boldsymbol{U}, \boldsymbol{S}) \leftarrow$ LOWRANKSYMAPPROX( ) ▷ Get $\hat{\boldsymbol{A}}_{\text{in}} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{U}^*$
  $(\boldsymbol{V}, \boldsymbol{D}) \leftarrow \texttt{eigs}(\boldsymbol{S}, r, \texttt{'lr'})$ ▷ Truncate full eigendecomposition
  $\boldsymbol{U} \leftarrow \boldsymbol{U}\boldsymbol{V}$ ▷ Consolidate orthonormal factors
  $\boldsymbol{D} \leftarrow \texttt{max}(\boldsymbol{D}, 0)$ ▷ Zero out negative eigenvalues
  return $(\boldsymbol{U}, \boldsymbol{D})$

If $\boldsymbol{A}$ is conjugate symmetric, then Corollary 5.3 and Proposition 6.1 show that $[\![\hat{\boldsymbol{A}}_{\text{sym}}]\!]_r$ admits an error bound identical to Corollary 6.4. We omit the details.

6.4. Fixed-rank positive-semidefinite approximation. Assume that the input matrix $\boldsymbol{A} \in \mathbb{H}^n_+$ is psd, and we wish to compute a rank-r psd approximation $[\![\hat{\boldsymbol{A}}_+]\!]_r$. First, form an initial approximation $\hat{\boldsymbol{A}}_+$ using the procedure (5.7) in subsection 5.4.2. Then compute an r-truncated positive eigenvalue decomposition of the matrix $\boldsymbol{S}$ defined in (5.5):

$$\boldsymbol{S} =: \boldsymbol{V}[\![\boldsymbol{D}_+]\!]_r\boldsymbol{V}^* \ + \ \text{approximation error.}$$

In view of the representation (5.7),

$$[\![\hat{\boldsymbol{A}}_+]\!]_r = (\boldsymbol{U}\boldsymbol{V})[\![\boldsymbol{D}_+]\!]_r(\boldsymbol{U}\boldsymbol{V})^*. \tag{6.6}$$

Algorithm 9 contains pseudocode for the fixed-rank psd approximation (6.6). The working storage is $\Theta(kn)$, and the arithmetic cost is $\Theta(k\ell n)$. If $\boldsymbol{A}$ is psd, then Corollary 5.3 and Proposition 6.1 show that $[\![\hat{\boldsymbol{A}}_{\text{psd}}]\!]_r$ satisfies an error bound identical to Corollary 6.4; we omit the details.

7. Computational experiments. This section presents the results of some numerical tests designed to evaluate the empirical performance of our sketching algorithms for low-rank matrix approximation. We demonstrate that the approximation quality improves when we impose structure, and we show that our theoretical parameter choices are effective. The presentation also includes comparisons with several other algorithms from the literature.

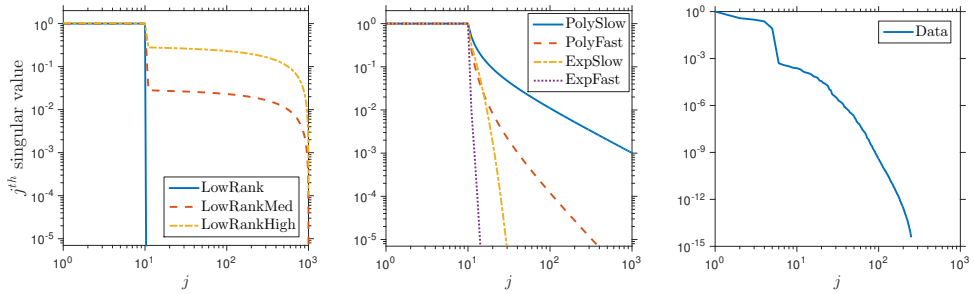


FIG. 1. Spectra of input matrices. *These plots display the singular value spectrum for an input matrix from each of the classes (`LowRank`, `LowRankMedNoise`, `LowRankHiNoise`, `PolyDecaySlow`, `PolyDecayFast`, `ExpDecaySlow`, `ExpDecayFast`, `Data`) described in subsection* 7.2.

7.1. Overview of experimental setup. For our numerical assessment, we work with the complex field ($\mathbb{F} = \mathbb{C}$). Results for the real field ($\mathbb{F} = \mathbb{R}$) are similar.

Let us summarize the procedure for studying the behavior of a specified approximation method on a given input matrix. Fix the input matrix $\boldsymbol{A}$ and the target rank r. Then select a pair (k, ℓ) of sketch size parameters where $k \geq r$ and $\ell \geq r$.

Each trial has the following form. We draw (complex) standard normal test matrices $(\boldsymbol{\Omega}, \boldsymbol{\Psi})$ to form the sketch $(\boldsymbol{Y}, \boldsymbol{W})$ of the input matrix. (We do not use the optional orthogonalization steps in Algorithm 1.) Next, we compute an approximation $\hat{\boldsymbol{A}}_{\text{out}}$ of the matrix $\boldsymbol{A}$ by means of a specified approximation algorithm. Then we calculate the error relative to the best rank-r approximation:

$$\text{relative error} \quad := \quad \frac{\|\boldsymbol{A} - \hat{\boldsymbol{A}}_{\text{out}}\|_{\text{F}}}{\tau_{r+1}(\boldsymbol{A})} - 1. \tag{7.1}$$

The tail energy τ_j is defined in (2.1). If $\hat{\boldsymbol{A}}_{\text{out}}$ is a rank-r approximation of $\boldsymbol{A}$, the relative error is always nonnegative. To facilitate comparisons, our experiments only examine fixed-rank approximation methods.

To obtain each data point, we repeat the procedure from the last paragraph 20 times, each time with the same input matrix $\boldsymbol{A}$ and an independent draw of the test matrices $(\boldsymbol{\Omega}, \boldsymbol{\Psi})$. Then we report the average relative error over the 20 trials.

We include our MATLAB implementations in the accompanying supplementary material file M111159_01.zip [local/web 78.3KB] for readers who seek more details on the methodology.

7.2. Classes of input matrices. We perform our numerical tests using several types of complex-valued input matrices. Figure 1 illustrates the singular spectrum of a matrix from each of the categories.

7.2.1. Synthetic examples. We fix a dimension parameter $n = 10^3$ and a parameter $R = 10$ that controls the rank of the "significant part" of the matrix. In our experiments, we compute approximations with target rank $r = 5$. Similar results hold when the parameter $R = 5$ and when $n = 10^4$.

We construct the following synthetic input matrices:

1. **Low-Rank + Noise:** These matrices take the form

$$\begin{bmatrix} \mathbf{I}_R & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad + \quad \sqrt{\frac{\gamma R}{2n^2}}(\boldsymbol{G} + \boldsymbol{G}^*) \quad \in \quad \mathbb{C}^{n \times n}.$$

The matrix $\boldsymbol{G}$ is complex standard normal. The quantity γ^{-2} can be interpreted as the signal-to-noise ratio (SNR). We consider three cases:

(a) *No noise* (`LowRank`): $\gamma = 0$.
(b) *Medium noise* (`LowRankMedNoise`): $\gamma = 10^{-2}$.
(c) *High noise* (`LowRankHiNoise`): $\gamma = 1$.

For these models, all the experiments are performed on a single exemplar that is drawn at random and then fixed.

2. **Polynomially decaying spectrum:** These matrices take the form

$$\operatorname{diag}\big(\underbrace{1,\ \ldots,\ 1}_{R},\ 2^{-p},\ 3^{-p},\ 4^{-p},\ \ldots,\ (n-R+1)^{-p}\big) \quad \in \quad \mathbb{C}^{n\times n},$$

where $p > 0$ controls the rate of decay. We consider two cases:

(a) *Slow polynomial decay* (`PolyDecaySlow`): $p = 1$.
(b) *Fast polynomial decay* (`PolyDecayFast`): $p = 2$.

3. **Exponentially decaying spectrum:** These matrices take the form

$$\operatorname{diag}\big(\underbrace{1,\ \ldots,\ 1}_{R},\ 10^{-q},\ 10^{-2q},\ 10^{-3q},\ \ldots,\ 10^{-(n-R)q}\big) \quad \in \quad \mathbb{C}^{n\times n},$$

where $q > 0$ controls the rate of decay. We consider two cases:

(a) *Slow exponential decay* (`ExpDecaySlow`): $q = 0.25$.
(b) *Fast exponential decay* (`ExpDecayFast`): $q = 1$.

We can focus on diagonal matrices because of the rotational invariance of the test matrices $(\boldsymbol{\Omega}, \boldsymbol{\Psi})$. Results for dense matrices are similar.

7.2.2. A matrix from an application in optimization. We also consider a dense, complex psd matrix (`Data`) obtained from a real-world phase-retrieval application. This matrix has dimension $n = 25{,}921$ and exact rank 250. The first five singular values decrease from 1 to around 0.1; there is a large gap between the fifth and sixth singular values; the remaining nonzero singular values decay very fast. See our paper [36] for more details about the role of sketching in this context.

7.3. Alternative sketching algorithms for matrix approximation. In addition to the algorithms we have presented, our numerical study comprises other methods that have appeared in the literature. We have modified all of these algorithms to improve their numerical stability and to streamline the computations. To the extent possible, we adopt the sketch (3.3) for all the algorithms to make their performance more comparable.

7.3.1. Methods based on the sketch (3.3). We begin with two additional methods that use the same sketch (3.3) as our algorithms.

First, let us describe a variant of a fixed-rank approximation scheme that was proposed by Woodruff [34, Thm. 4.3, display 2]. First, form a matrix product and compute its orthogonal–triangular factorization: $\boldsymbol{\Psi Q} =: \boldsymbol{UT}$, where $\boldsymbol{U} \in \mathbb{F}^{\ell\times k}$ has orthonormal columns. Then construct the rank-r approximation

$$\hat{\boldsymbol{A}}_{\text{woo}} := \boldsymbol{QT}^{\dagger}[\![\boldsymbol{U}^{*}\boldsymbol{W}]\!]_{r}. \tag{7.2}$$

Woodruff shows that $\hat{\boldsymbol{A}}_{\text{woo}}$ satisfies (1.5) when the sketch size scales as $k = \Theta(r/\varepsilon)$ and $\ell = \Theta(k/\varepsilon^2)$. Compare this result with Remark 6.5.

Second, we outline a fixed-rank approximation method that is implicit in Cohen et al. [12, sect. 10.1]. First, compute the r dominant left singular vectors of the range sketch: $(\boldsymbol{V}, \sim, \sim) := \texttt{svds}(\boldsymbol{Y}, r)$. Form a matrix product and compute its orthogonal–triangular factorization: $\boldsymbol{\Psi V} =: \boldsymbol{UT}$, where $\boldsymbol{U} \in \mathbb{F}^{\ell \times r}$. Then form the rank-$r$ approximation

$$\hat{\boldsymbol{A}}_{\text{cemmp}} := \boldsymbol{V}\boldsymbol{T}^{\dagger}[\![\boldsymbol{U}^*\boldsymbol{W}]\!]_r. \tag{7.3}$$

The results of Cohen et al. imply that $\hat{\boldsymbol{A}}_{\text{cemmp}}$ satisfies (1.5) when the sketch size scales as $k = \Theta(r/\varepsilon^2)$ and $\ell = \Theta(r/\varepsilon^2)$.

The approximations (7.2) and (7.3) both appear similar to our fixed-rank approximation, Algorithm 7. Nevertheless, they are derived from other principles, and their behavior is noticeably different.

7.3.2. A method based on an extended sketch. Next, we present a variant of a recent approach that requires a more complicated sketch and more elaborate computations. The following procedure is adapted from [6, Thm. 12], using simplifications suggested in [33, sect. 3].

Let $\boldsymbol{A} \in \mathbb{F}^{m \times n}$ be an input matrix, and let r be a target rank. Choose integer parameters k and s that satisfy $r \le k \le s \le \min\{m, n\}$. For consistent notation, we also introduce a redundant parameter $\ell = k$. Draw and fix *four* test matrices:

$$\boldsymbol{\Psi} \in \mathbb{F}^{k \times m}; \quad \boldsymbol{\Omega} \in \mathbb{F}^{n \times \ell}; \quad \boldsymbol{\Phi} \in \mathbb{F}^{s \times m}; \quad \text{and} \quad \boldsymbol{\Xi} \in \mathbb{F}^{n \times s}. \tag{7.4}$$

The matrices $(\boldsymbol{\Psi}, \boldsymbol{\Omega})$ are standard normal, while $(\boldsymbol{\Phi}, \boldsymbol{\Xi})$ are SRFTs; see subsection 3.9. The sketch now has *three* components:

$$\boldsymbol{W} := \boldsymbol{\Psi A}; \quad \boldsymbol{Y} := \boldsymbol{A\Omega}; \quad \text{and} \quad \boldsymbol{Z} := \boldsymbol{\Phi A \Xi}. \tag{7.5}$$

To store the test matrices and the sketch, we need $(2k+1)(m+n)+s(s+2)$ numbers.

To obtain a rank-r approximation of the input matrix $\boldsymbol{A}$, first compute four thin orthogonal–triangular factorizations:

$$\begin{aligned} \boldsymbol{Y} &=: \boldsymbol{Q}_1\boldsymbol{R}_1 \qquad \text{and} \qquad \boldsymbol{W} =: \boldsymbol{R}_2^*\boldsymbol{Q}_2^*; \\ \boldsymbol{\Phi Q}_1 &=: \boldsymbol{U}_1\boldsymbol{T}_1 \qquad \text{and} \qquad \boldsymbol{Q}_2^*\boldsymbol{\Xi} =: \boldsymbol{T}_2^*\boldsymbol{U}_2^*. \end{aligned}$$

Then construct the rank-r approximation

$$\hat{\boldsymbol{A}}_{\text{bwz}} := \boldsymbol{Q}_1\boldsymbol{T}_1^{\dagger}[\![\boldsymbol{U}_1^*\boldsymbol{Z}\boldsymbol{U}_2]\!]_r(\boldsymbol{T}_2^*)^{\dagger}\boldsymbol{Q}_2^*. \tag{7.6}$$

By adapting and correcting [6, Thm. 12], one can show that $\hat{\boldsymbol{A}}_{\text{bwz}}$ achieves (1.5) for sketch size parameters that satisfy $k = \Theta(r/\varepsilon)$ and $s = \Theta((r\log(1+r))^2/\varepsilon^6)$. With this scaling, the total storage cost for the random matrices and the sketch is $\Theta((m+n)r/\varepsilon + (r\log(1+r))^2/\varepsilon^6)$.

The authors of [6] refer to their method as "optimal" because the scaling of the term $(m+n)r/\varepsilon$ in the storage cost cannot be improved [9, Thm. 4.10]. Nevertheless, because of the ε^{-6} term, the bound is incomparable with the storage costs achieved by other algorithms.

7.4. Performance with oracle parameter choices. It is challenging to compare the relative performance of sketching algorithms for matrix approximation because of the theoretical nature of previous research. In particular, earlier work does not offer any practical guidance for selecting the sketch size parameters.

The only way to make a fair comparison is to study the *oracle performance* of the algorithms. That is, for each method, we fix the total storage, and we determine the minimum relative error that the algorithm can achieve. This approach allows us to see which techniques are most promising for further development. Nevertheless, we must emphasize that the oracle performance is not achievable in practice.

7.4.1. Computing the oracle error. It is straightforward to compare our fixed-rank approximation methods, Algorithms 7 to 9, with the alternatives (7.2) and (7.3) from the literature. In each case, the sketch (3.3) requires storage of $(k+\ell)(m+n)$ numbers, so we can parameterize the cost by $T := k + \ell$. For a given choice of T, we obtain the oracle performance by minimizing the empirical approximation error for each algorithm over all pairs (k, ℓ) where the sum $k + \ell = T$.

It is trickier to include the Boutsidis–Woodruff–Zhong [6, Thm. 12] method (7.6). For a given T, we obtain the oracle performance of (7.6) by minimizing the empirical approximation error over pairs (k, s) for which the storage cost of the sketch (7.5) matches the cost of the simple sketch (3.3). That is, $(2k + 1)(m + n) + s(s + 2) \approx T(m + n)$.

7.4.2. Numerical comparison with prior work. For each input matrix described in subsection 7.2, Figure 2 compares the oracle performance of our fixed-rank approximation, Algorithm 7, against several alternative methods, (7.2), (7.3), and (7.6), from the literature. We make the following observations:

- For matrices that are well approximated by a low-rank matrix (`LowRank`, `PolyDecayFast`, `ExpDecaySlow`, `ExpDecayFast`, `Data`), our fixed-rank approximation, Algorithm 7, dominates all other methods when the storage budget is adequate. In particular, for the rank-1 approximation of the matrix `Data`, our approach achieves relative errors 3–6 orders of magnitude better than any competitor.
- When we consider matrices that are poorly approximated by a low-rank matrix (`LowRankMedNoise`, `LowRankHiNoise`, `PolyDecaySlow`), the recent method (7.6) of Boutsidis, Woodruff, and Zhong [6, Thm. 12] has the best performance, especially when the storage budget is small. But see subsection 7.4.3 for more texture.
- Our method, Algorithm 7, performs reliably for all of the input matrices, and it is the only method that can achieve high accuracy for the matrix `Data`. Its behavior is less impressive for matrices that have poor low-rank approximations (`LowRankMedNoise`, `LowRankHiNoise`, `PolyDecaySlow`), but it is still competitive for these examples.
- The method (7.6) of Boutsidis, Woodruff, and Zhong [6, Thm. 12] offers mediocre performance for matrices with good low-rank approximations (`LowRank`, `ExpDecaySlow`, `ExpDecayFast`, `Data`). Strikingly, this approach fails to produce a high-accuracy rank-5 approximation of the rank-10 matrix `LowRank`, even with a large storage budget.
- The method (7.2) of Woodruff [34, Thm. 4.3, display 2] is competitive for most synthetic examples, but it performs rather poorly on the matrix `Data`.
- The method (7.3) of Cohen et al. [12, sect. 10.1] has the worst performance for almost all the examples.

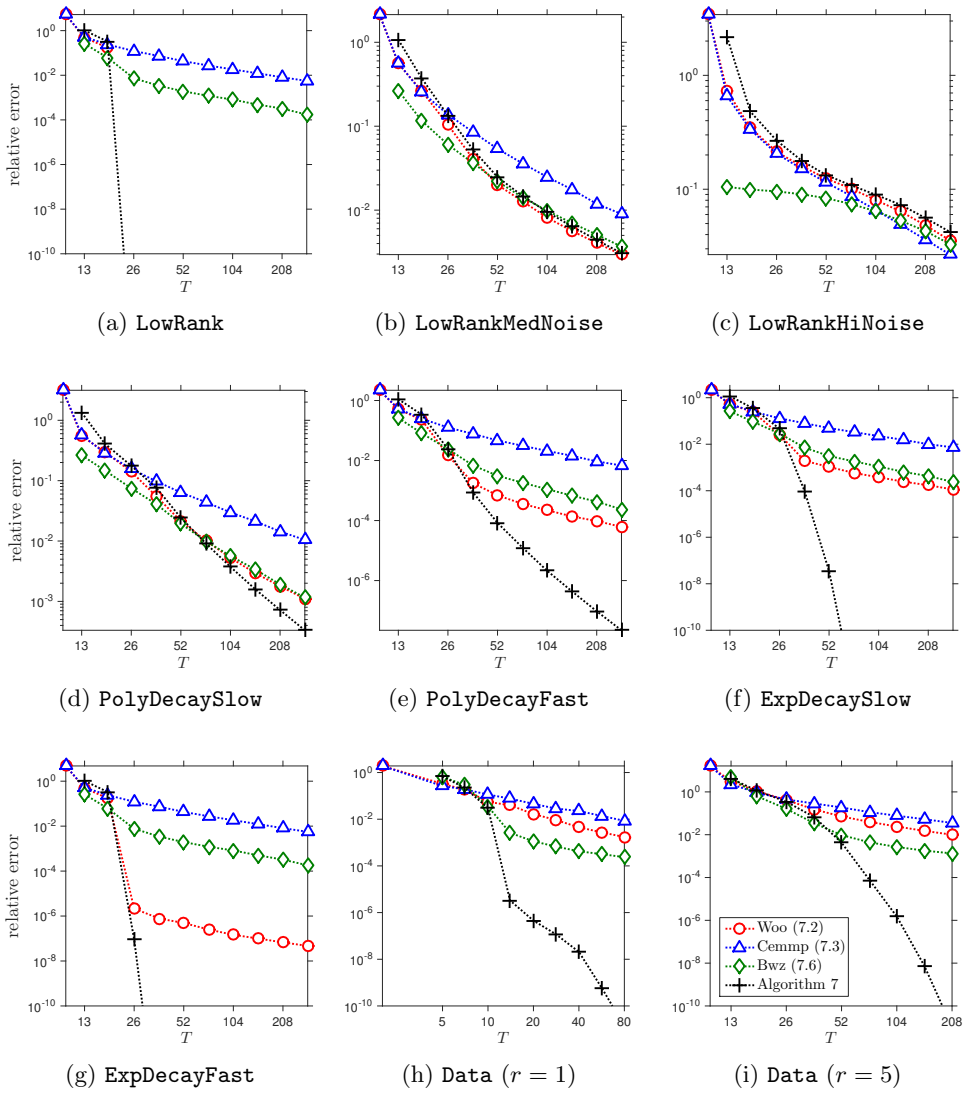


FIG. 2. Oracle performance of sketching algorithms for fixed-rank matrix approximation as a function of storage cost. *For each of the input matrices described in subsection* 7.2, *we compare the oracle performance of our fixed-rank approximation, Algorithm* 7, *against alternative methods* (7.2), (7.3), *and* (7.6) *from the literature. The matrix dimensions are* $m = n = 10^3$ *for the synthetic examples and* $m = n = 25{,}921$ *for the matrix* `Data` *from the phase retrieval application. Each approximation has rank* $r = 5$, *unless otherwise stated. The variable* T *on the horizontal axis is (proportional to) the total storage used by each sketching method. Each data series displays the best relative error* (7.1) *that the specified algorithm can achieve with storage* T. *See subsection* 7.4.1 *for details.*

In summary, Algorithm 7 has the best all-around behavior, while the Boutsidis–Woodruff–Zhong [6, Thm. 12] method (7.6) works best for matrices that have a poor low-rank approximation. See subsection 7.6 for more discussion.

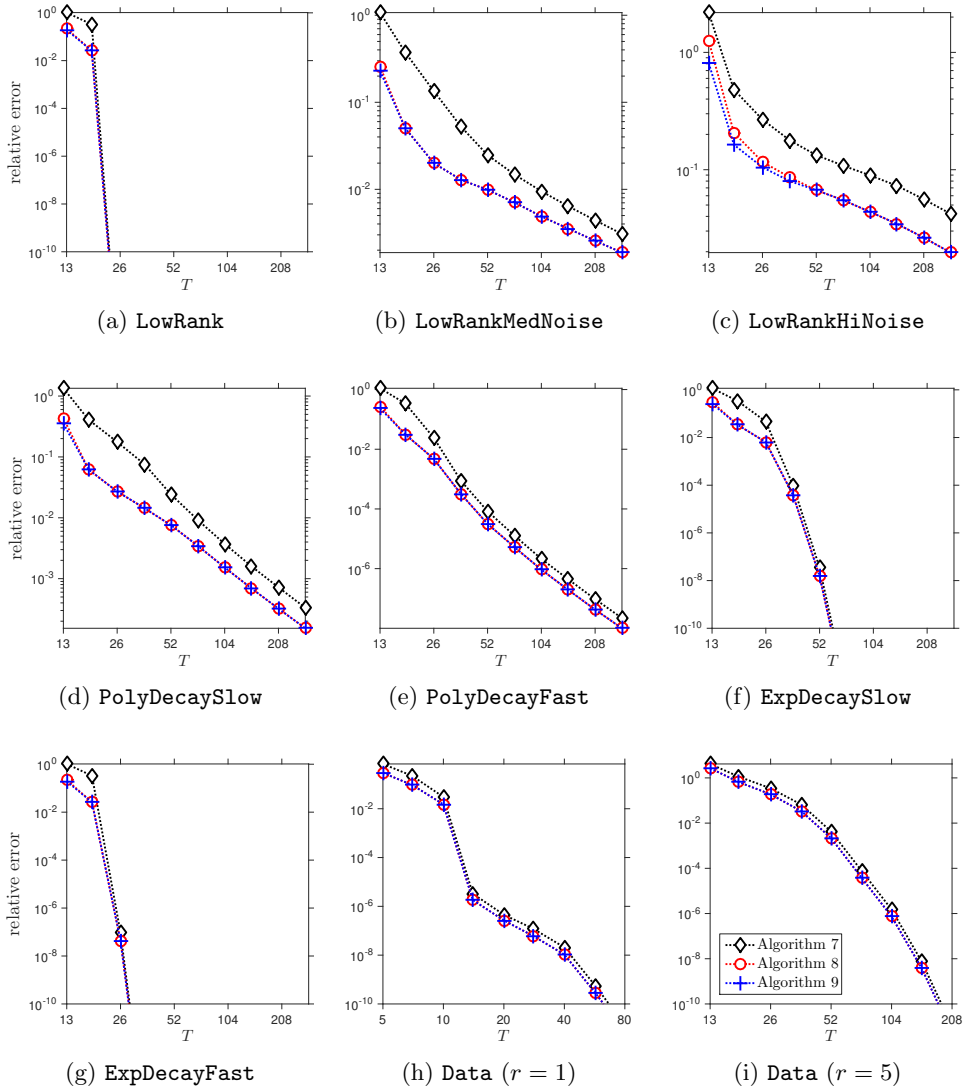


FIG. 3. Oracle performance of sketching algorithms for structured fixed-rank matrix approximation as a function of storage cost. *For each of the input matrices described in subsection* 7.2, *we compare the oracle performance of the unstructured approximation (Algorithm* 7*), the conjugate symmetric approximation (Algorithm* 8*), and the psd approximation (Algorithm* 9*). The matrix dimensions are* $m = n = 10^3$ *for the synthetic examples and* $m = n = 25,921$ *for the matrix* `Data` *from the phase retrieval application. Each approximation has rank* $r = 5$*, unless otherwise stated. The variable* T *on the horizontal axis is (proportional to) the total storage used by each sketching method. Each data series displays the best relative error* (7.1) *that the specified algorithm can achieve with storage* T*. See subsection* 7.4.1 *for details.*

7.4.3. Structured approximations. In this section, we investigate the effect of imposing structure on the low-rank approximations. Figure 3 compares the oracle performance of our fixed-rank approximation methods, Algorithms 7 to 9. We make the following observations:

- The symmetric approximation method, Algorithm 8, and the psd approximation method, Algorithm 9, are very similar to each other for all examples.
- The structured approximations, Algorithms 8 and 9, always improve upon the unstructured approximation, Algorithm 7. The benefit is most significant for matrices that have a poor low-rank approximation (`LowRankMedNoise`, `LowRankHiNoise`, `PolyDecaySlow`).
- Algorithms 8 and 9 match or exceed the performance of the Boutsidis–Woodruff–Zhong [6, Thm. 12] method (7.6) for all examples.

In summary, if we know that the input matrix has structure, we can achieve a decisive advantage by enforcing the structure in the approximation.

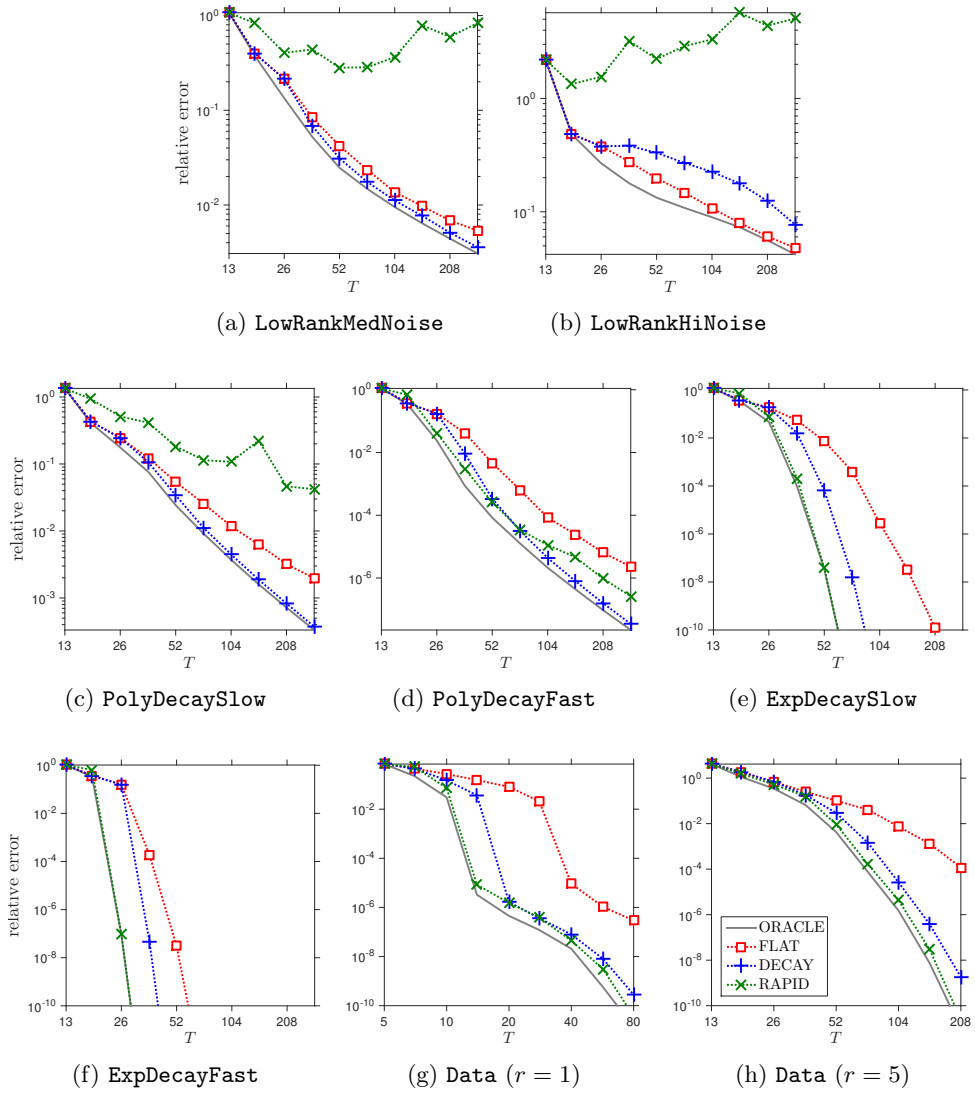
7.5. Performance with theoretical parameter choices. It remains to understand how closely we can match the oracle performance of Algorithms 7 to 9 in practice. To that end, we must choose the sketch size parameters a priori using only the knowledge of the target rank r and the total sketch size T. In some instances, we may also have insight into the spectral decay of the input matrix. Figure 4 shows how the fixed-rank approximation method, Algorithm 7, performs with the theoretical parameter choices outlined in subsection 4.5. We make the following observations:

- The parameter recommendation (4.7), designed for a matrix with a flat spectral tail, works well for the matrices `LowRankMedNoise`, `LowRankHiNoise`, and `PolyDecaySlow`. We also learn that this parameter choice should not be used for matrices with spectral decay.
- The parameter recommendation (4.9), for a matrix with a slowly decaying spectrum, is suited to the examples `LowRankMedNoise`, `LowRankHiNoise`, `PolyDecaySlow`, and `PolyDecayFast`. This parameter choice is effective for the remaining examples as well.
- The parameter recommendation (4.10), for a matrix with a rapidly decaying spectrum, is appropriate for the examples `PolyDecayFast`, `ExpDecaySlow`, `ExpDecayFast`, and `Data`. This choice must not be used unless the spectrum decays quickly.
- We have observed that the same parameter recommendations allow us to achieve near-oracle performance for the structured matrix approximations, Algorithms 8 and 9. As in the unstructured case, it helps if we tune the parameter choice to the type of input matrix.

In summary, we always achieve reasonably good performance using the parameter choice (4.9). Furthermore, if we match the parameter selection (4.7), (4.9), and (4.10) to the spectral properties of the input matrix, we can almost achieve the oracle performance in practice.

7.6. Recommendations. Among the fixed-rank approximation methods that we studied, the most effective are Algorithms 7 to 9 and the Boutsidis–Woodruff–Zhong [6, Thm. 12] method (7.6). Let us make some final observations based on our numerical experience.

Algorithms 7 to 9 are superior to methods from the literature for input matrices that have good low-rank approximations. Although Algorithm 7 suffers when the input matrix has a poor low-rank approximation, the structured variants, Algorithms 8 and 9, match or exceed other algorithms for all the examples we tested. We have



(a) `LowRankMedNoise` (b) `LowRankHiNoise`

(c) `PolyDecaySlow` (d) `PolyDecayFast` (e) `ExpDecaySlow`

(f) `ExpDecayFast` (g) `Data` ($r = 1$) (h) `Data` ($r = 5$)

FIG. 4. Performance of a sketching algorithm for fixed-rank matrix approximation with a priori parameter choices. *For each of the input matrices described in subsection* 7.2, *we compare the oracle performance of the fixed-rank approximation, Algorithm* 7, *against its performance at theoretically motivated parameter choices. The matrix dimensions are* $m = n = 10^3$ *for the synthetic examples and* $m = n = 25{,}921$ *for the matrix* `Data` *from the phase retrieval application. Each approximation has rank* $r = 5$, *unless otherwise stated. The variable* T *on the horizontal axis is (proportional to) the total storage used by each sketching method. The oracle performance is drawn from Figure* 2. *Each data series displays the relative error* (7.1) *that Algorithm* 7 *achieves for a specific parameter selection. The parameter choice* `FLAT` (4.7) *is designed for matrices with a flat spectral tail;* `DECAY` (4.9) *is for a slowly decaying spectrum;* `RAPID` (4.10) *is for a rapidly decaying spectrum. See subsection* 7.5 *for details.*

also established that we can attain near-oracle performance for our methods using the a priori parameter recommendations from subsection 4.5. Finally, our methods are simple and easy to implement.

The Boutsidis–Woodruff–Zhong [6, Thm. 12] method (7.6) exhibits the best per-

formance for matrices that have very poor low-rank approximations when the storage budget is very small. This benefit is diminished by its mediocre performance for matrices that do admit good low-rank approximations. The method (7.6) requires more complicated sketches and additional computation. Unfortunately, the analysis in [6] does not provide guidance on implementation.

In conclusion, we recommend the sketching methods, Algorithms 7 to 9, for computing structured low-rank approximations. In future research, we will try to design new methods that simultaneously dominate our algorithms and (7.6).

Appendix A. Analysis of the low-rank approximation. In this appendix, we develop theoretical results on the performance of the basic low-rank approximation (4.3) implemented in Algorithms 3 and 4.

A.1. Facts about random matrices. Our arguments require classical formulae for the expectations of functions of a standard normal matrix. In the real case, these results are [19, Props. A.1 and A.6]. The complex case follows from the same principles, so we omit the details.

FACT A.1. *Let $\boldsymbol{G} \in \mathbb{F}^{t\times s}$ be a standard normal matrix. For all matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ with conforming dimensions,*

$$\mathbb{E}\,\|\boldsymbol{BGC}\|_{\mathrm{F}}^2 = \beta\|\boldsymbol{B}\|_{\mathrm{F}}^2\|\boldsymbol{C}\|_{\mathrm{F}}^2. \tag{A.1}$$

Furthermore, if $t > s + \alpha$,

$$\mathbb{E}\,\|\boldsymbol{G}^\dagger\|_{\mathrm{F}}^2 = \frac{1}{\beta}\cdot\frac{s}{t-s-\alpha} = \frac{1}{\beta}\cdot f(s,t). \tag{A.2}$$

The numbers α and β are given by (2.2)*; the function f is introduced in* (2.3).

A.2. Results from randomized linear algebra. Our arguments also depend heavily on the analysis of randomized low-rank approximation developed in [19, sect. 10]. We state these results using the familiar notation from sections 3 and 4.

FACT A.2 (Halko, Martinsson, and Tropp [19]). *Fix $\boldsymbol{A} \in \mathbb{F}^{m\times n}$. Let ϱ be a natural number such that $\varrho < k - \alpha$. Draw the random test matrix $\boldsymbol{\Omega} \in \mathbb{F}^{k\times n}$ from the standard normal distribution. Then the matrix $\boldsymbol{Q}$ computed by* (4.1) *satisfies*

$$\mathbb{E}_{\boldsymbol{\Omega}}\,\|\boldsymbol{A} - \boldsymbol{QQ}^*\boldsymbol{A}\|_{\mathrm{F}}^2 \le (1 + f(\varrho,k))\cdot\tau_{\varrho+1}^2(\boldsymbol{A}).$$

The number α is given by (2.2)*; the function f is introduced in* (2.3).

This result follows immediately from the proof of [19, Thm. 10.5] using Fact A.1 to handle both the real and complex cases simultaneously.

A.3. Proof of Theorem 4.3: Frobenius error bound. In this section, we establish a second Frobenius-norm error bound for the low-rank approximation (4.3). We maintain the notation from sections 3 and 4, and we state explicitly when we are making distributional assumptions on the test matrices.

A.3.1. Decomposition of the approximation error. Fact A.2 formalizes the intuition that $\boldsymbol{A} \approx \boldsymbol{Q}(\boldsymbol{Q}^*\boldsymbol{A})$. The main object of the proof is to demonstrate that $\boldsymbol{X} \approx \boldsymbol{Q}^*\boldsymbol{A}$. The first step in the argument is to break down the approximation error into these two parts.

LEMMA A.3. *Let $\boldsymbol{A}$ be an input matrix, and let $\hat{\boldsymbol{A}} = \boldsymbol{QX}$ be the approximation defined in* (4.3). *The approximation error decomposes as*

$$\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}}^2 = \|\boldsymbol{A} - \boldsymbol{QQ}^*\boldsymbol{A}\|_{\mathrm{F}}^2 + \|\boldsymbol{X} - \boldsymbol{Q}^*\boldsymbol{A}\|_{\mathrm{F}}^2.$$

We omit the proof, which is essentially just the Pythagorean theorem.

A.3.2. Approximating the second factor. Next, we develop an explicit expression for the error in the approximation $\boldsymbol{X} \approx \boldsymbol{Q}^*\boldsymbol{A}$. It is convenient to construct a matrix $\boldsymbol{P} \in \mathbb{F}^{n\times(n-k)}$ with orthonormal columns that satisfies

$$\boldsymbol{P}\boldsymbol{P}^* = \mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*. \tag{A.3}$$

Introduce the matrices

$$\boldsymbol{\Psi}_1 := \boldsymbol{\Psi}\boldsymbol{P} \in \mathbb{F}^{\ell\times(n-k)} \quad \text{and} \quad \boldsymbol{\Psi}_2 := \boldsymbol{\Psi}\boldsymbol{Q} \in \mathbb{F}^{\ell\times k}. \tag{A.4}$$

We are now prepared to state the result.

LEMMA A.4. *Assume that the matrix $\boldsymbol{\Psi}_2$ has full column rank. Then*

$$\boldsymbol{X} - \boldsymbol{Q}^*\boldsymbol{A} = \boldsymbol{\Psi}_2^\dagger\boldsymbol{\Psi}_1(\boldsymbol{P}^*\boldsymbol{A}). \tag{A.5}$$

The matrices $\boldsymbol{\Psi}_1$ and $\boldsymbol{\Psi}_2$ are defined in (A.4).

Proof. Recall that $\boldsymbol{W} = \boldsymbol{\Psi}\boldsymbol{A}$, and calculate that

$$\boldsymbol{W} = \boldsymbol{\Psi}\boldsymbol{A} = \boldsymbol{\Psi}\boldsymbol{P}\boldsymbol{P}^*\boldsymbol{A} + \boldsymbol{\Psi}\boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A} = \boldsymbol{\Psi}_1(\boldsymbol{P}^*\boldsymbol{A}) + \boldsymbol{\Psi}_2(\boldsymbol{Q}^*\boldsymbol{A}).$$

The second relation holds because $\boldsymbol{P}\boldsymbol{P}^* + \boldsymbol{Q}\boldsymbol{Q}^* = \mathbf{I}$. Then we use (A.4) to identify $\boldsymbol{\Psi}_1$ and $\boldsymbol{\Psi}_2$. By hypothesis, the matrix $\boldsymbol{\Psi}_2$ has full column rank, so we can left-multiply the last display by $\boldsymbol{\Psi}_2^\dagger$ to attain

$$\boldsymbol{\Psi}_2^\dagger\boldsymbol{W} = \boldsymbol{\Psi}_2^\dagger\boldsymbol{\Psi}_1(\boldsymbol{P}^*\boldsymbol{A}) + \boldsymbol{Q}^*\boldsymbol{A}.$$

Turning back to (4.2), we identify $\boldsymbol{X} = \boldsymbol{\Psi}_2^\dagger\boldsymbol{W}$. ∎

A.3.3. The expected Frobenius-norm error in the second factor. We are now prepared to compute the average Frobenius-norm error in approximating $\boldsymbol{Q}^*\boldsymbol{A}$ by means of the matrix $\boldsymbol{X}$. In contrast to the previous steps, this part of the argument relies on distributional assumptions on the test matrix $\boldsymbol{\Psi}$. Remarkably, for a Gaussian test matrix, $\boldsymbol{X}$ is even an unbiased estimator of the factor $\boldsymbol{Q}^*\boldsymbol{A}$.

LEMMA A.5. *Assume that $\boldsymbol{\Psi} \in \mathbb{F}^{\ell\times n}$ is a standard normal matrix that is independent from $\boldsymbol{\Omega}$. Then*

$$\mathbb{E}_{\boldsymbol{\Psi}}[\boldsymbol{X} - \boldsymbol{Q}^*\boldsymbol{A}] = \mathbf{0}.$$

Furthermore,

$$\mathbb{E}_{\boldsymbol{\Psi}}\,\|\boldsymbol{X} - \boldsymbol{Q}^*\boldsymbol{A}\|_{\mathrm{F}}^2 = f(k,\ell)\cdot\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\|_{\mathrm{F}}^2.$$

Proof. Observe that $\boldsymbol{P}$ and $\boldsymbol{Q}$ are partial isometries with orthogonal ranges. Owing to the marginal property of the standard normal distribution, the random matrices $\boldsymbol{\Psi}_1$ and $\boldsymbol{\Psi}_2$ are statistically independent standard normal matrices. In particular, $\boldsymbol{\Psi}_2 \in \mathbb{F}^{\ell\times k}$ almost surely has full column rank because (3.1) requires that $\ell \geq k$.

First, take the expectation of the identity (A.5) to see that

$$\mathbb{E}_{\boldsymbol{\Psi}}[\boldsymbol{X} - \boldsymbol{Q}^*\boldsymbol{A}] = \mathbb{E}_{\boldsymbol{\Psi}_2}\,\mathbb{E}_{\boldsymbol{\Psi}_1}[\boldsymbol{\Psi}_2^\dagger\boldsymbol{\Psi}_1\boldsymbol{P}^*\boldsymbol{A}] = \mathbf{0}.$$

In the first relation, we use the statistical independence of $\boldsymbol{\Psi}_1$ and $\boldsymbol{\Psi}_2$ to write the expectation as an iterated expectation. Then we observe that $\boldsymbol{\Psi}_1$ is a matrix with zero mean.

Next, take the expected squared Frobenius norm of (A.5) to see that

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\Psi}} \|\boldsymbol{X} - \boldsymbol{Q}^*\boldsymbol{A}\|_{\mathrm{F}}^2 &= \mathbb{E}_{\boldsymbol{\Psi}_2} \mathbb{E}_{\boldsymbol{\Psi}_1} \|\boldsymbol{\Psi}_2^\dagger \boldsymbol{\Psi}_1(\boldsymbol{P}^*\boldsymbol{A})\|_{\mathrm{F}}^2 \\ &= \beta \cdot \mathbb{E}_{\boldsymbol{\Psi}_2} \left[\|\boldsymbol{\Psi}_2^\dagger\|_{\mathrm{F}}^2 \cdot \|\boldsymbol{P}^*\boldsymbol{A}\|_{\mathrm{F}}^2\right] = f(k,\ell) \cdot \|\boldsymbol{P}^*\boldsymbol{A}\|_{\mathrm{F}}^2.\end{aligned}$$

The last two identities follow from (A.1) and (A.2), respectively, where we use the fact that $\boldsymbol{\Psi}_2 \in \mathbb{F}^{\ell \times k}$. To conclude, note that

$$\|\boldsymbol{P}^*\boldsymbol{A}\|_{\mathrm{F}}^2 = \|\boldsymbol{P}\boldsymbol{P}^*\boldsymbol{A}\|_{\mathrm{F}}^2 = \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\|_{\mathrm{F}}^2.$$

The first relation holds because $\boldsymbol{P}$ is a partial isometry and the Frobenius norm is unitarily invariant. Last, we apply the definition (A.3) of $\boldsymbol{P}$. ∎

A.3.4. Proof of Theorem 4.3. We are now prepared to complete the proof of the Frobenius-norm error bound stated in Theorem 4.3. For this argument, we assume that the test matrices $\boldsymbol{\Omega} \in \mathbb{F}^{n \times k}$ and $\boldsymbol{\Psi} \in \mathbb{F}^{\ell \times m}$ are drawn independently from the standard normal distribution.

According to Lemma A.3,

$$\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}}^2 = \|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\|_{\mathrm{F}}^2 + \|\boldsymbol{X} - \boldsymbol{Q}^*\boldsymbol{A}\|_{\mathrm{F}}^2.$$

Take the expectation of the last display to reach

$$\begin{aligned}\mathbb{E}\,\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}}^2 &= \mathbb{E}_{\boldsymbol{\Omega}}\,\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\|_{\mathrm{F}}^2 + \mathbb{E}_{\boldsymbol{\Omega}}\,\mathbb{E}_{\boldsymbol{\Psi}}\,\|\boldsymbol{X} - \boldsymbol{Q}^*\boldsymbol{A}\|_{\mathrm{F}}^2 \\ &= (1 + f(k,\ell)) \cdot \mathbb{E}_{\boldsymbol{\Omega}}\,\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\|_{\mathrm{F}}^2 \\ &\le (1 + f(k,\ell)) \cdot (1 + f(\varrho,k)) \cdot \tau_{\varrho+1}^2(\boldsymbol{A}).\end{aligned}$$

In the first line, we use the independence of the two random matrices to write the expectation as an iterated expectation. To reach the second line, we apply Lemma A.5 to the second term. Invoke the randomized linear algebra result, Fact A.2. Finally, minimize over eligible indices $\varrho < k - \alpha$.

REFERENCES

[1] N. Ailon and B. Chazelle, *Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform*, in STOC'06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing, ACM, New York, 2006, pp. 557–563, https://doi.org/10.1145/1132516.1132597.

[2] N. Ailon and B. Chazelle, *The fast Johnson–Lindenstrauss transform and approximate nearest neighbors*, SIAM J. Comput., 39 (2009), pp. 302–322, https://doi.org/10.1137/060673096.

[3] J. Bourgain, S. Dirksen, and J. Nelson, *Toward a unified theory of sparse dimensionality reduction in Euclidean space*, Geom. Funct. Anal., 25 (2015), pp. 1009–1088, https://doi.org/10.1007/s00039-015-0332-9.

[4] C. Boutsidis, D. Garber, Z. Karnin, and E. Liberty, *Online principal components analysis*, in Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 2015, pp. 887–901.

[5] C. Boutsidis and A. Gittens, *Improved matrix algorithms via the subsampled randomized Hadamard transform*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1301–1340, https://doi.org/10.1137/120874540.

[6] C. Boutsidis, D. Woodruff, and P. Zhong, *Optimal principal component analysis in distributed and streaming models*, in Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC 2016), Cambridge, MA, 2016.

[7] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge University Press, Cambridge, 2004, https://doi.org/10.1017/CBO9780511804441.

[8] K. Clarkson and D. Woodruff, *Low-rank PSD approximation in input-sparsity time*, Unpublished, Jan. 2017.

[9] K. L. Clarkson and D. P. Woodruff, *Numerical linear algebra in the streaming model*, in Proceedings of the 41st ACM Symposium on Theory of Computing (STOC), ACM, New York, 2009, pp. 205–214.

[10] K. L. Clarkson and D. P. Woodruff, *Low rank approximation and regression in input sparsity time*, in STOC'13—Proceedings of the 2013 ACM Symposium on Theory of Computing, ACM, New York, 2013, pp. 81–90, https://doi.org/10.1145/2488608.2488620.

[11] M. Cohen, *Nearly tight oblivious subspace embeddings by trace inequalities*, in Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, Philadelphia, ACM, New York, 2016, pp. 278–287.

[12] M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu, *Dimensionality reduction for k-means clustering and low rank approximation*, in Proceedings of the 47th Annual ACM Symposium on Theory of Computing, ACM, New York, 2015, pp. 163–172.

[13] M. B. Cohen, J. Nelson, and D. P. Woodruff, *Optimal approximate matrix product in terms of stable rank*, in the 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016), I. Chatzigiannakis, M. Mitzenmacher, Y. Rabani, and D. Sangiorgi, eds., Leibniz International Proceedings in Informatics (LIPIcs) 55, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2016, pp. 11:1–11:14, https://doi.org/10.4230/LIPIcs.ICALP.2016.11.

[14] J. Demmel, I. Dumitriu, and O. Holtz, *Fast linear algebra is stable*, Numer. Math., 108 (2007), pp. 59–91, https://doi.org/10.1007/s00211-007-0114-x.

[15] D. Feldman, M. Schmidt, and C. Sohler, *Turning big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering*, in Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, ACM, New York, 2012, pp. 1434–1453.

[16] D. Feldman, M. Volkov, and D. Rus, *Dimensionality reduction of massive sparse datasets using coresets*, in Advances in Neural Information Processing Systems 29 (NIPS 2016), Curran Associates, 2016, pp 2766–2774.

[17] A. Frieze, R. Kannan, and S. Vempala, *Fast Monte-Carlo algorithms for finding low-rank approximations*, J. ACM, 51 (2004), pp. 1025–1041, https://doi.org/10.1145/1039488.1039494.

[18] M. Gu, *Subspace iteration randomization and singular value problems*, SIAM J. Sci. Comput., 37 (2015), pp. A1139–A1173, https://doi.org/10.1137/130938700.

[19] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288, https://doi.org/10.1137/090771806.

[20] N. J. Higham, *Matrix nearness problems and applications*, in Applications of Matrix Theory (Bradford, 1988), Oxford University Press, New York, 1989, pp. 1–27.

[21] P. Jain, C. Jin, S. M. Kakade, P. Netrapalli, and A. Sidford, *Streaming PCA: Matching matrix Bernstein and near-optimal finite sample guarantees for Oja's algorithm*, in 29th Annual Conference on Learning Theory, Columbia University, New York, 2016, pp. 1147–1164.

[22] H. Li, G. C. Linderman, A. Szlam, K. P. Stanton, Y. Kluger, and M. Tygert, *Algorithm 971: An implementation of a randomized algorithm for principal component analysis*, ACM Trans. Math. Softw., 43 (2017), pp. 28:1–28:14, http://doi.acm.org/10.1145/3004053.

[23] Y. Li, H. L. Nguyen, and D. P. Woodruff, *Turnstile streaming algorithms might as well be linear sketches*, in STOC'14—Proceedings of the 2014 ACM Symposium on Theory of Computing, ACM, New York, 2014, pp. 174–183.

[24] E. Liberty, *Accelerated Dense Random Projections*, Ph.D. thesis, Yale University, New Haven, CT, 2009.

[25] M. W. Mahoney, *Randomized algorithms for matrices and data*, Found. Trends Machine Learn., 3 (2011), pp. 123–224.

[26] P.-G. Martinsson, V. Rokhlin, and M. Tygert, *A randomized algorithm for the decomposition of matrices*, Appl. Comput. Harmon. Anal., 30 (2011), pp. 47–68, https://doi.org/10.1016/j.acha.2010.02.003.

[27] X. Meng and M. W. Mahoney, *Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression*, in STOC'13—Proceedings of the 2013 ACM Symposium on Theory of Computing, ACM, New York, 2013, pp. 91–100, https://doi.org/10.1145/2488608.2488621.

[28] J. Nelson and H. L. Nguyen, *OSNAP: faster numerical linear algebra algorithms via sparser subspace embeddings*, in 2013 IEEE 54th Annual Symposium on Foundations of Computer

Science—FOCS 2013, IEEE Computer Soc., Los Alamitos, CA, 2013, pp. 117–126, https://doi.org/10.1109/FOCS.2013.21.

[29] J. Nelson and H. L. Nguyen, *Lower bounds for oblivious subspace embeddings*, in Automata, Languages, and Programming. Part I, Lecture Notes in Comput. Sci. 8572, Springer, Heidelberg, 2014, pp. 883–894, https://doi.org/10.1007/978-3-662-43948-7_73.

[30] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, *Latent semantic indexing: a probabilistic analysis*, J. Comput. System Sci., 61 (2000), pp. 217–235, https://doi.org/10.1006/jcss.2000.1711. Special issue on the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (Seattle, WA, 1998).

[31] J. A. Tropp, *Improved analysis of the subsampled randomized Hadamard transform*, Adv. Adapt. Data Anal., 3 (2011), pp. 115–126, https://doi.org/10.1142/S1793536911000787.

[32] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Randomized Single-View Algorithms for Low-Rank Matrix Approximation*, ACM Report 2017-01, Caltech, Pasadena, CA, 2017, https://arXiv.org/abs/1609.00048v1.

[33] J. Upadhyay, *Fast and Space-Optimal Low-Rank Factorization in the Streaming Model with Application in Differential privacy*, preprint, https://arXiv.org/abs/1604.01429, 2016.

[34] D. P. Woodruff, *Sketching as a tool for numerical linear algebra*, Found. Trends Theoret. Comput. Sci., 10 (2014), pp. iv+157.

[35] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, *A fast randomized algorithm for the approximation of matrices*, Appl. Comput. Harmon. Anal., 25 (2008), pp. 335–366.

[36] A. Yurtsever, M. Udell, J. A. Tropp, and V. Cevher, *Sketchy decisions: Convex low-rank matrix optimization with optimal storage*, in Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, 2017, http://proceedings.mlr.press/v54/yurtsever17a/yurtsever17a.pdf.